目录

[1. SQL 1](#_Toc2675)

[2. Node & Express without the views 4](#_Toc11249)

[2.1 init the project 4](#_Toc10787)

[2.2 step 1&2:setting up express and adding middleware 5](#_Toc28279)

[2.3 config debugger to test if the program works 6](#_Toc28656)

[2.4 step 3: add data models 6](#_Toc7831)

[2.1 Controller’s Problems 7](#_Toc24094)

[2.2 Add Viewer 7](#_Toc20229)

[2.3 link views to controller 10](#_Toc30730)

[2.4 test form data from the login page 12](#_Toc11071)

[2.5 test JWT token with postman 14](#_Toc25221)

[2.6 add auth to all routes except login page 15](#_Toc30449)

[2.7 connect login frontpage with api 15](#_Toc17447)

[2.8 add fileupload function: routes and controller 17](#_Toc20871)

[2.9 Node upload to IPFS 17](#_Toc19731)

# SQL

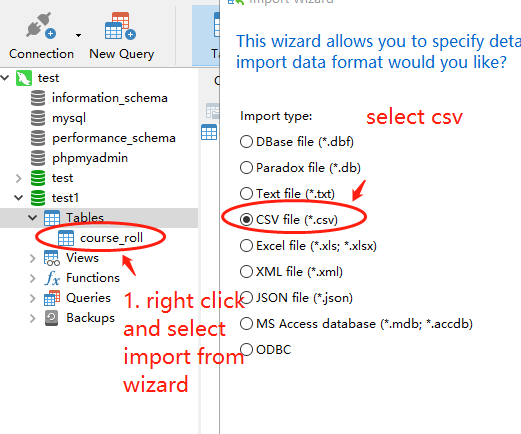
* 1. Prerequisite:

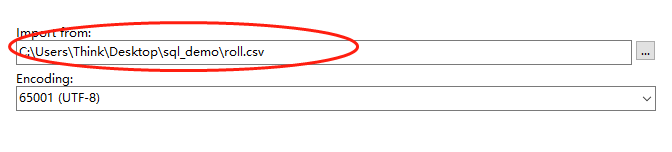
install xampp: <https://www.apachefriends.org/index.html>

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

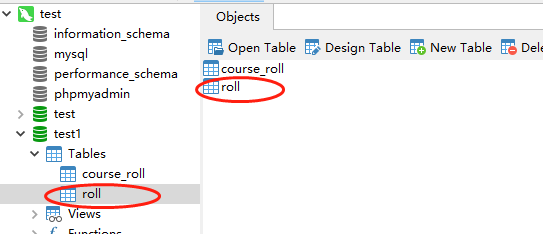
Data sets: <https://github.com/onebit256/sql_demo>

* 1. import data sets



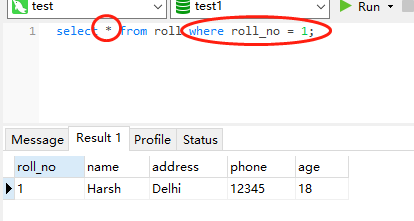


Select the right path, and keep clicking next til the end, then you will see the imported data base



* 1. insert, select, delete, joins

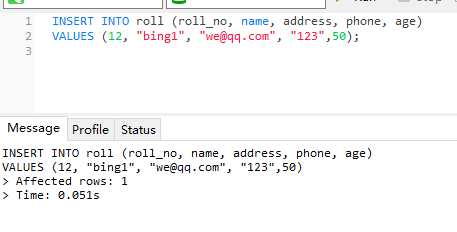
Select

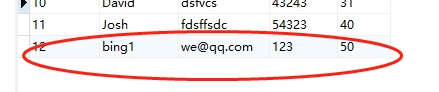


- \* means display all fields in the selected table

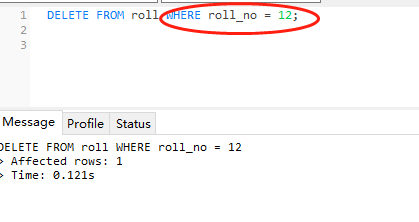
- “Where” is the condition to filter the records

Insert

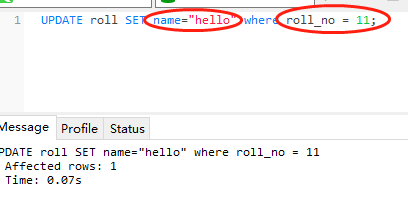


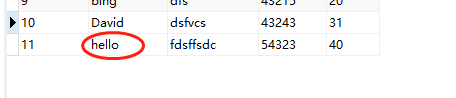


Delete

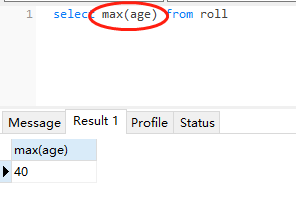


Update

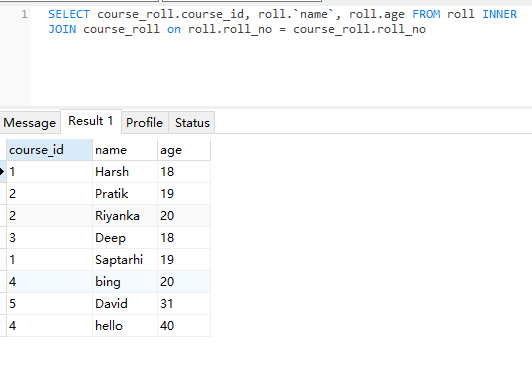




Max



Inner join



# Node & Express without the views

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

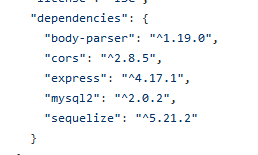
Use auth from prject 2: <https://github.com/onebit256/Node-js-JWT-auth---Medium>

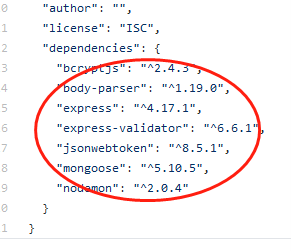
Final Code: <https://github.com/onebit256/batch2-lab4-node-without-views>

## init the project

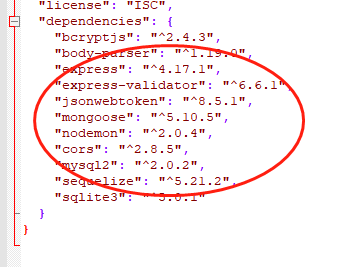
Npm init

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



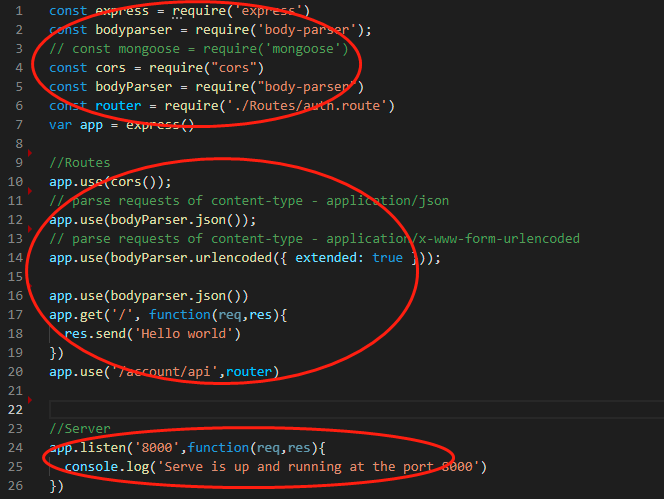


To the package.json in the new project folder



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

Copy index.js from project 2 to app.js

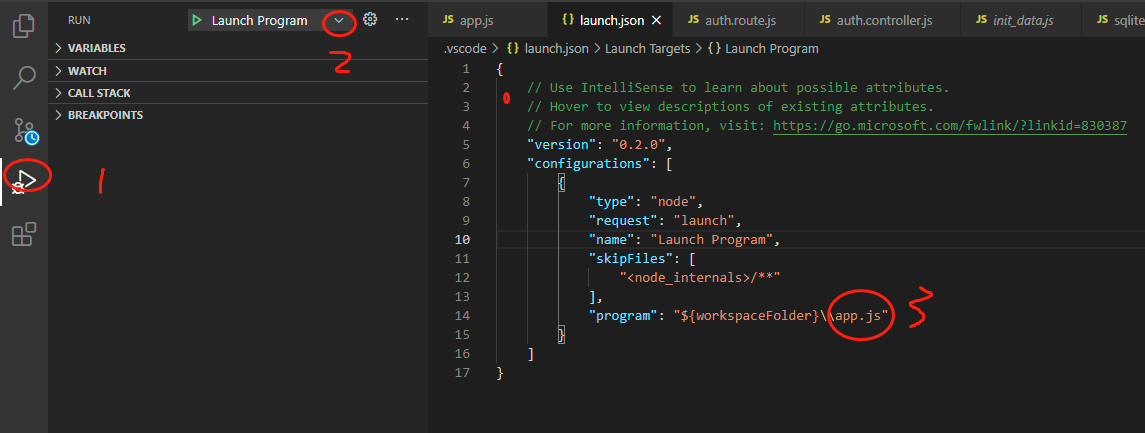


1 create express app

2 add middle ware

3 add port listener

## config debugger to test if the program works



Click 1

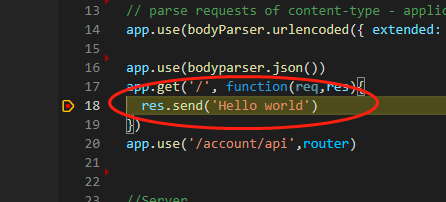
Click 2 select nodejs

Replace index.js with app.js

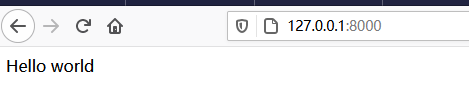
Set break point on app.js

When type 127.0.0.1:8000 on the browser

The break points stops



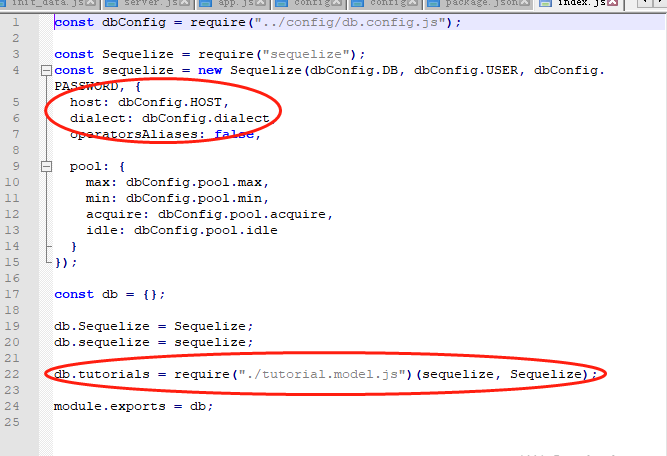
So the router works



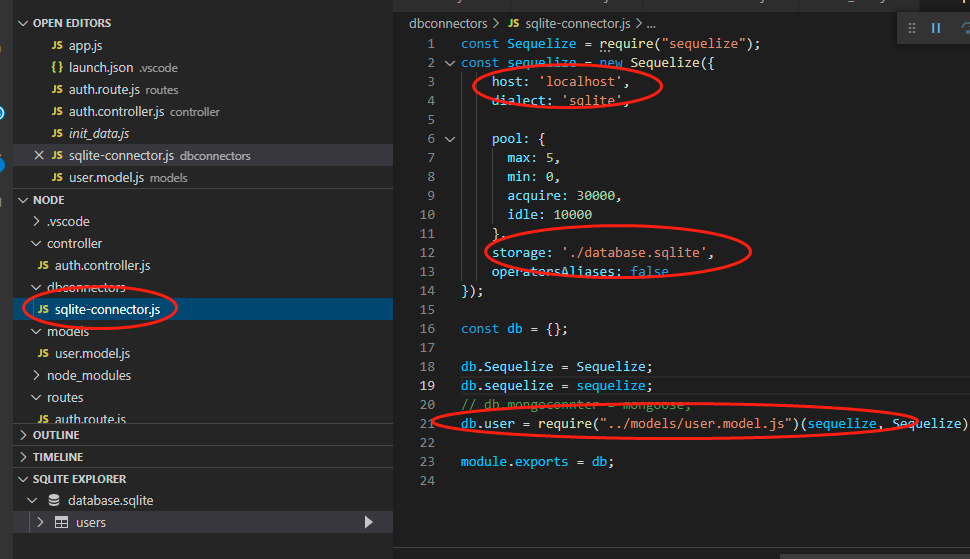
## step 3: add data models

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

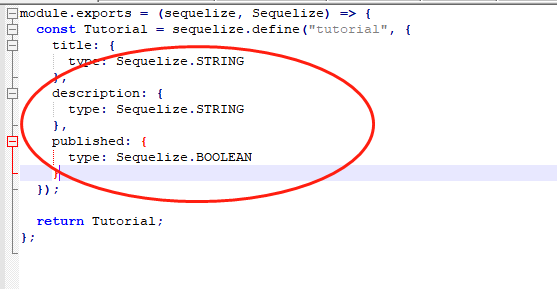
Replace mysql config to sqlite config



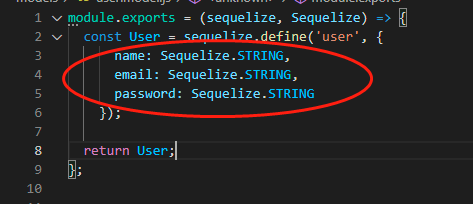
To



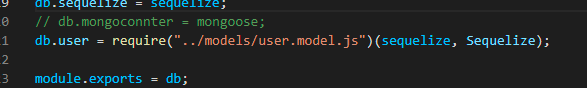
Then copy tutorial.model.js under model folder from project 1 to the new project under the model folder as well. And set user model



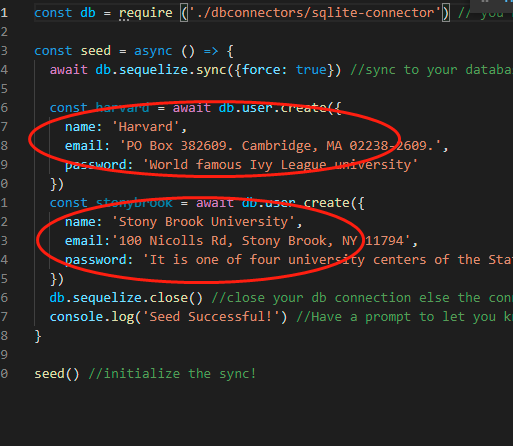
To



Then add user element to the db in dbconnetor

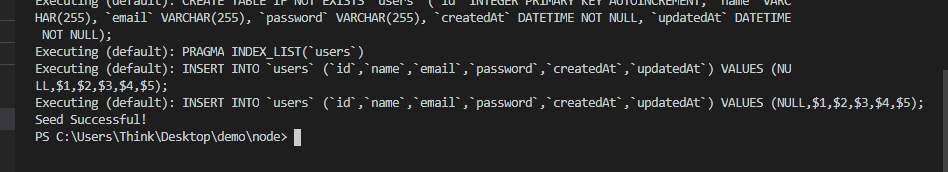


Create a init\_data.js to test if the dbconnetor works

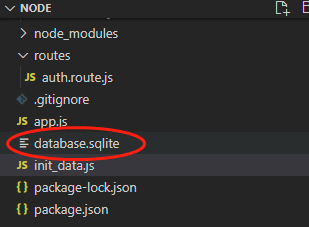


Node init\_data.js

Console prints

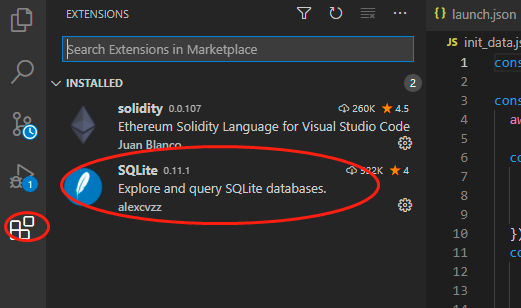


And



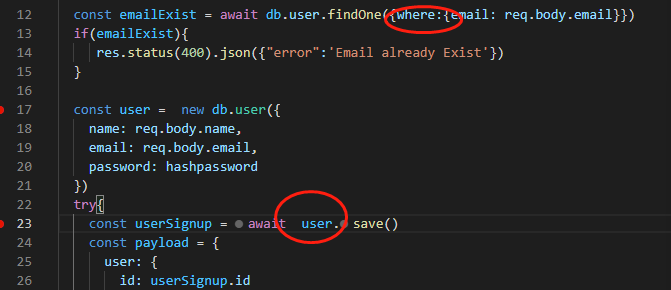
Database.sqlite is generated

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



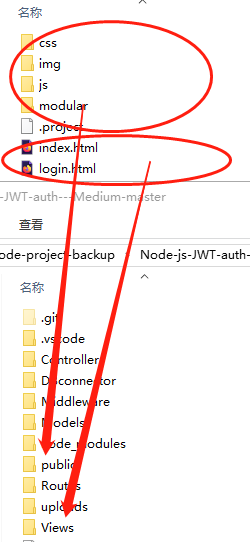
After install Sqlite add-on

## 2.1 Controller’s Problems

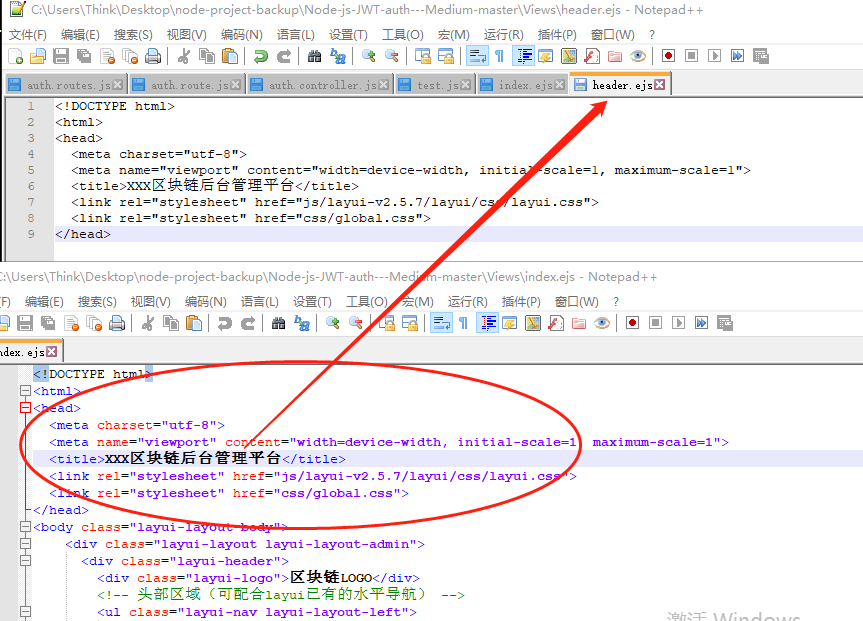


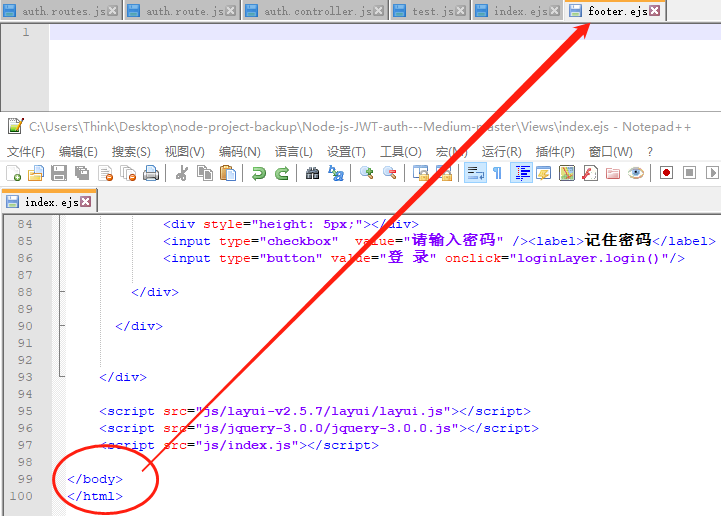
## 2.2 Add Viewer

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

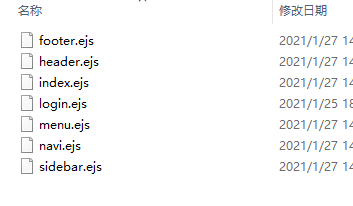


2 split html into parts

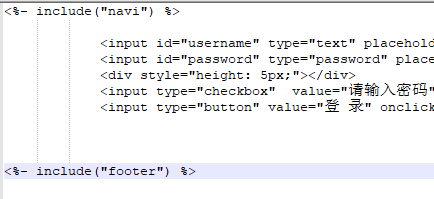




Eventually:



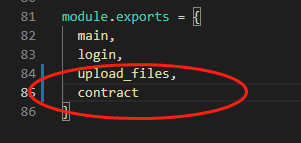
Index.ejs

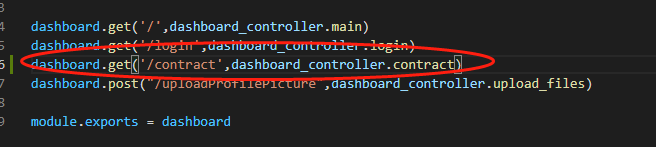


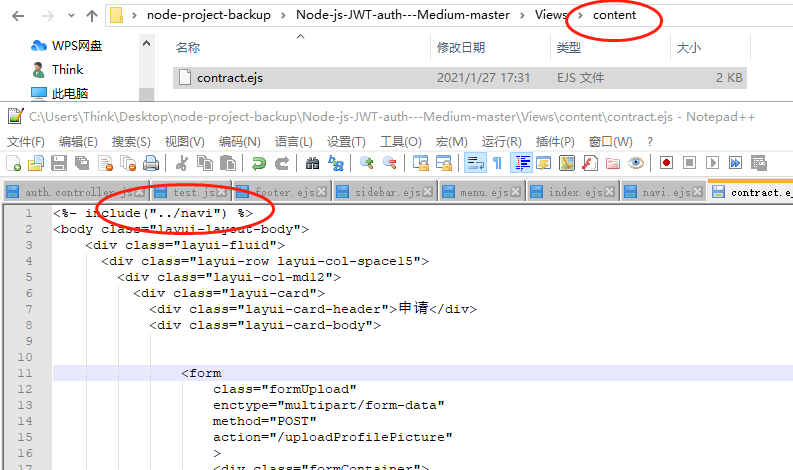
## 2.3 link views to controller

1. Add controller to apply\_for

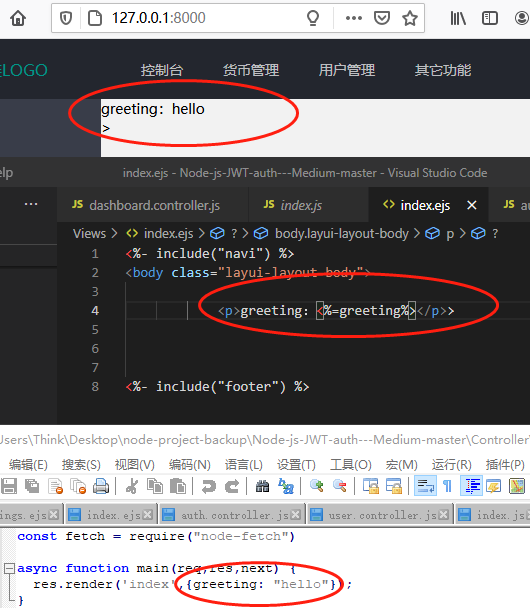






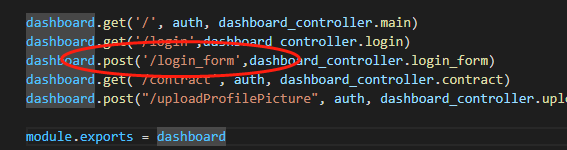


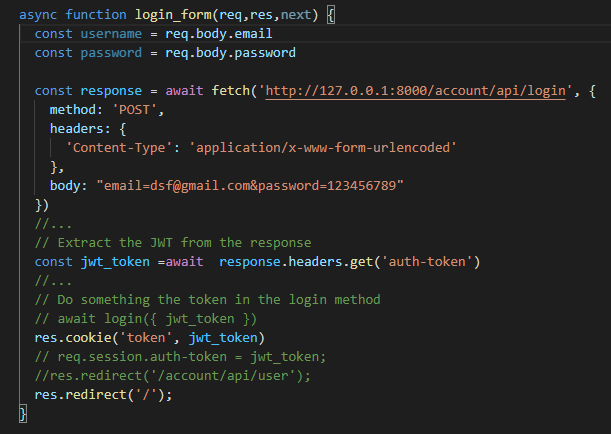
2 pass data from controller to the views



## 2.4 test form data from the login page

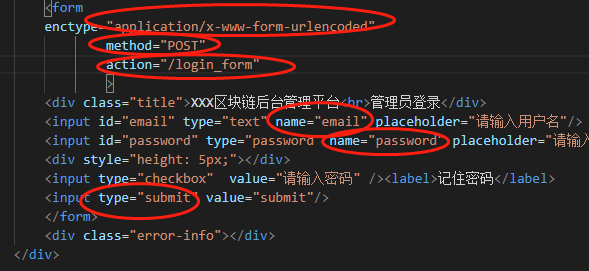
1 create a route to handle the form data, and create its corresponding controller



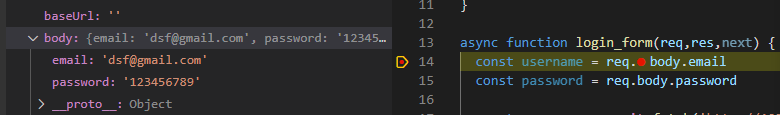


1. Create form

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

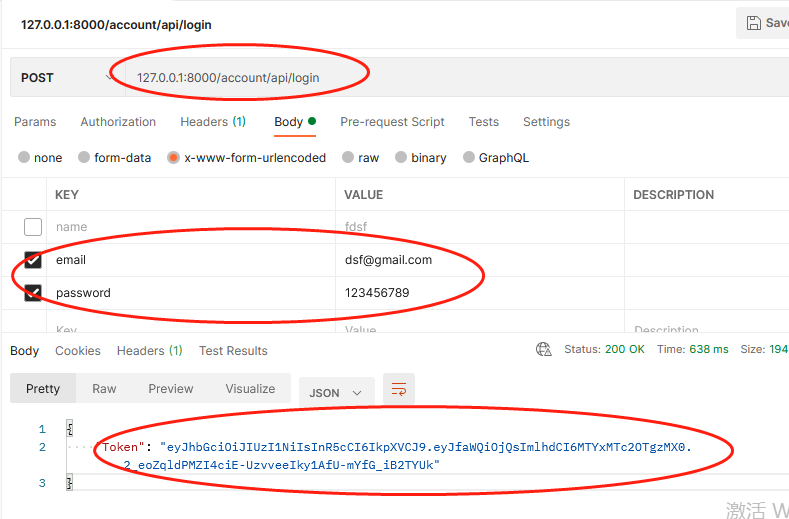


1. Test

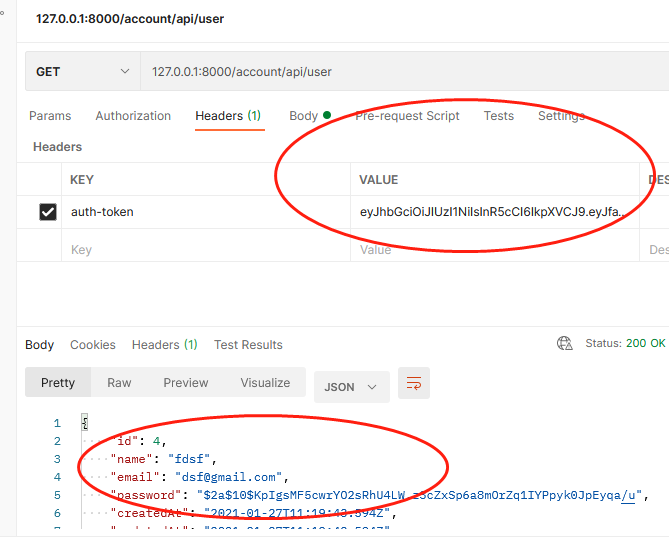


## 2.5 test JWT token with postman

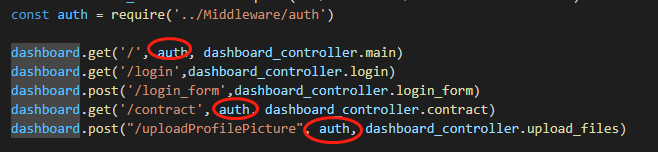
1. Get token



1. Add this token to get header

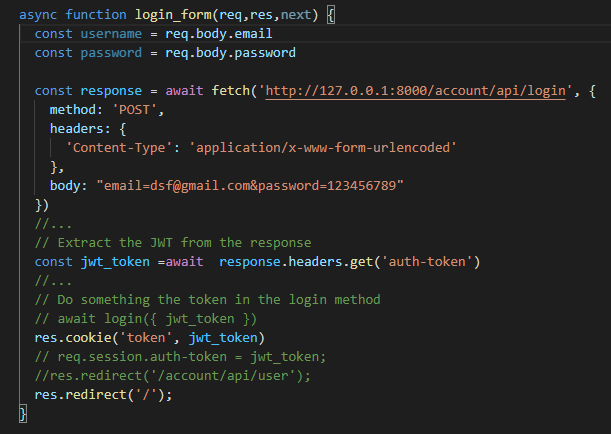


## 2.6 add auth to all routes except login page



## 2.7 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

## 2.8 add fileupload function: routes and controller

## 2.9 Node upload to IPFS