Content

[1 Add views to node 1](#_Toc15459)

[1.1 Integrating html 1](#_Toc30786)

[1.2 link views to controller 5](#_Toc31969)

[2 Pass data from controller to the views 6](#_Toc29953)

[3 Pass form data to login and signup 7](#_Toc11930)

[3.1 signup 7](#_Toc16398)

[3.2 login 10](#_Toc9321)

[4 add auth middleware to all routes except login page 12](#_Toc29414)

[5 login use cookie 13](#_Toc946)

[6 fileupload 14](#_Toc20406)

# 1 Add views to node

Static html repo: <https://github.com/onebit256/static-html>

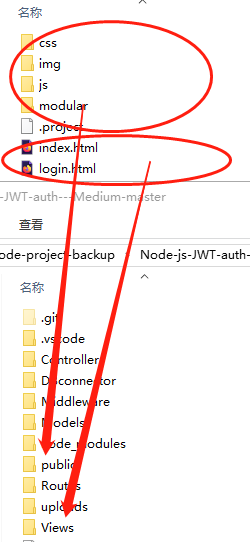
Node without-views: <https://github.com/onebit256/batch2-lab4-node-without-views>

Tasks:

1. Turn this static html into nodejs template
2. Add the views to the nodejs app and turn this nodejs api application to a normal nodejs application with views

## 1.1 Integrating html

1. Create a public folder in the project older. And move css js to that public folder,
2. Create a vewis folder and change html to ejs and move them to the views folder
3. Npm i ejs



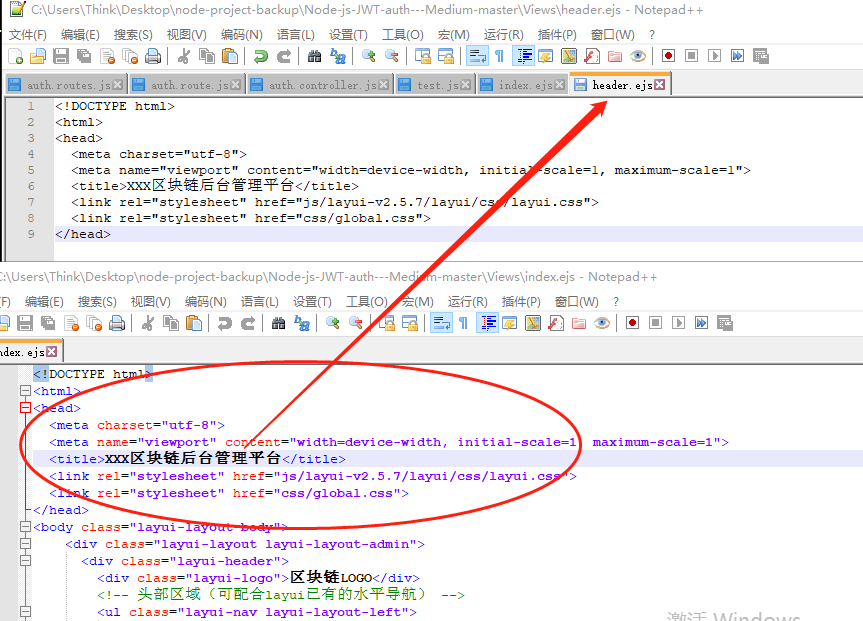
3 split html into parts

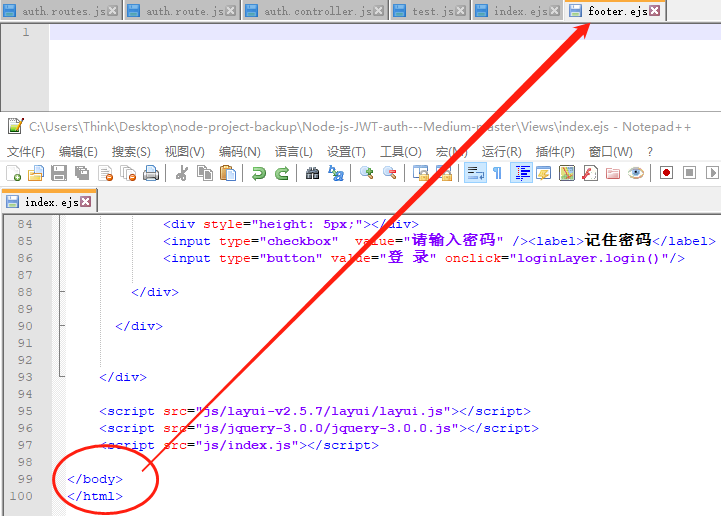


How to split?

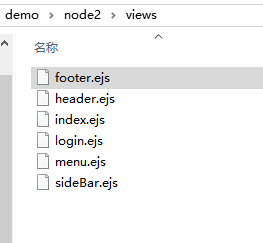
1 always write header and footer

2 in this case, we can create a component for the sidebar, create a component for the menu, and a component for the main content, if necessary.

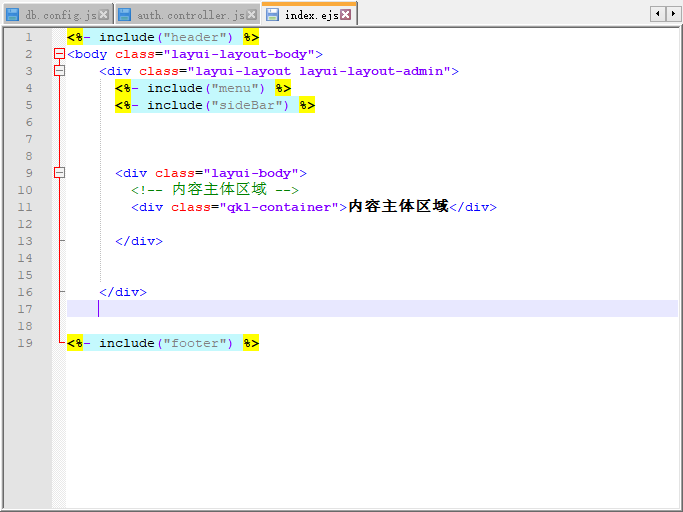




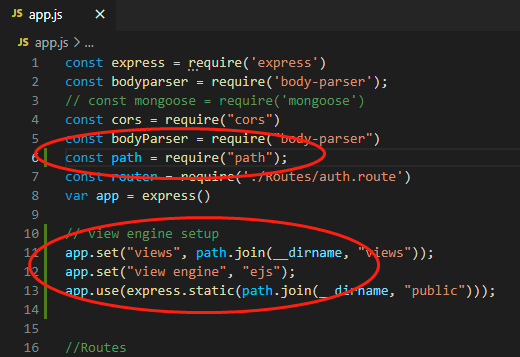
Eventually:



Index.ejs



4 link the template to nodejs program: add the following 3 lines to app.js

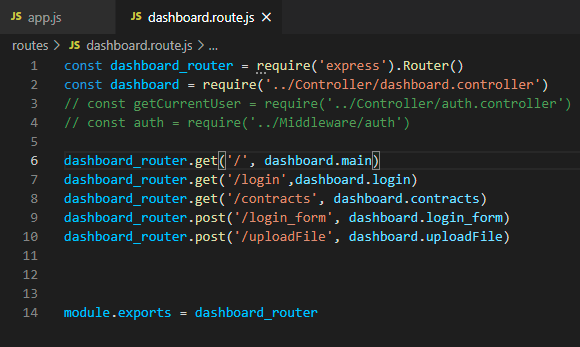


## 1.2 link views to controller

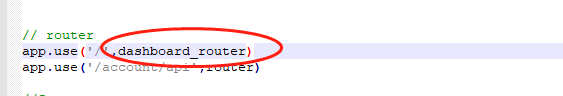
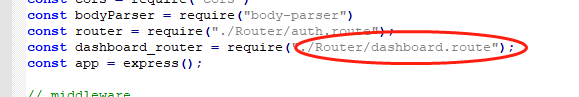
What we want to do in this section is add a set of views, routes and controllers, and allow users to do registration via the views instead of api requests

1. Define routes

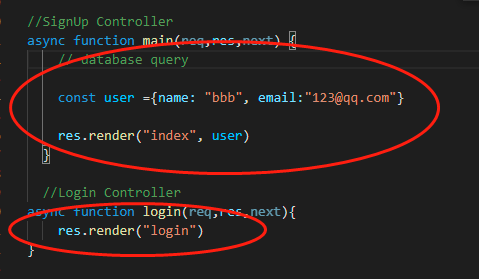
Create a file called dashboard.route.js

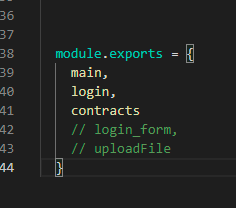


1. Add this route to app use

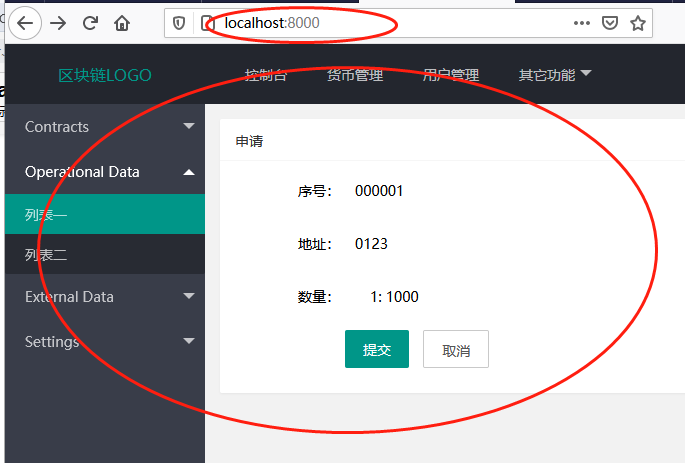


1. Add controller



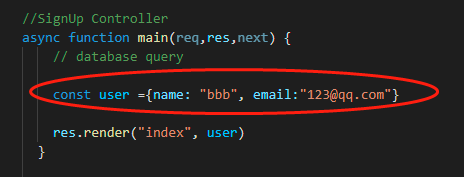


Verify if the views connected. Run debug, you will be able to see:



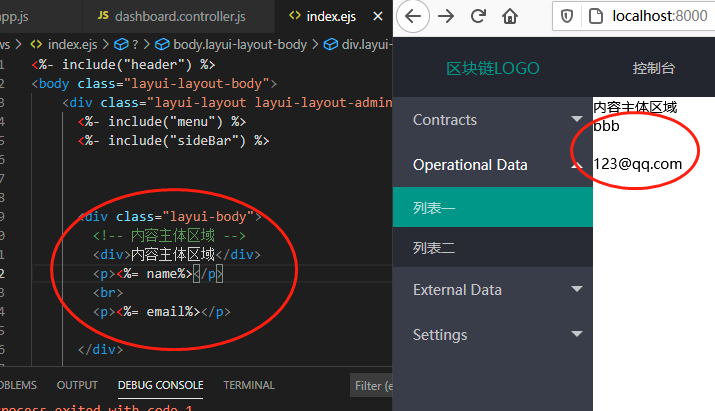
# 2 Pass data from controller to the views

* 1. defines the data in controller



Res.render sends the data to the ejs views

* 1. sets parameters in views



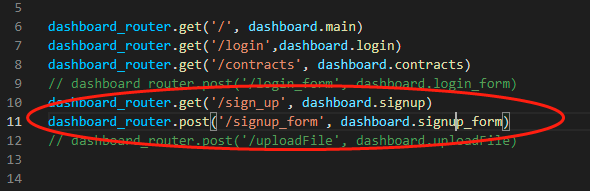
2.3 open the browser, you will see the data passed down to the view

# 3 Pass form data to login and signup

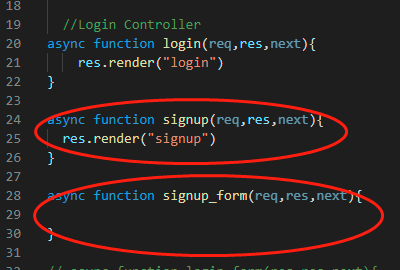
In this section, we need to write a set of model, view, and controller for login and signup

## 3.1 signup

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

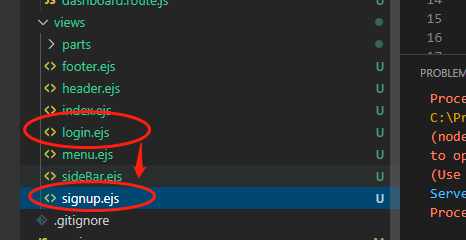


2 add controller



add controller for the sign\_up and for the signup form

3 create view for sign up



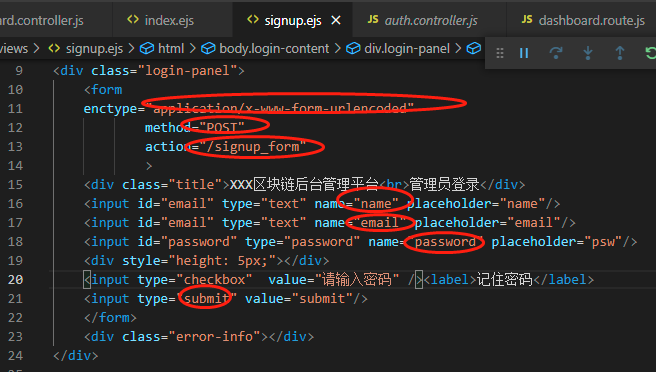
Copy login.ejs directly and change the name to signup.ejs

3 Create form in the view

Change the view from



to



Notes:

- Set form data to www-form-urlencode

- Assign the route path to the action and change the name to “/signup\_form”

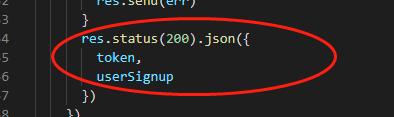
- Set the variable names

- Set the submit button type to be “submit”

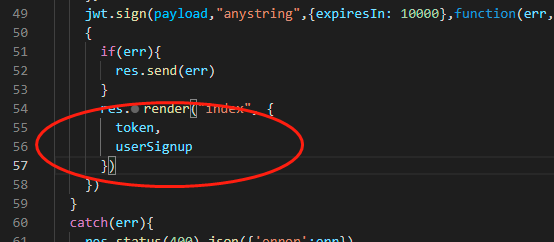
4 copy the sign up logic from the auth.controller.js to signup\_form controller

Why? Because auth controller logic is for api. We need to write a set of mvc, and their logic is the same, so we could copy the code directly

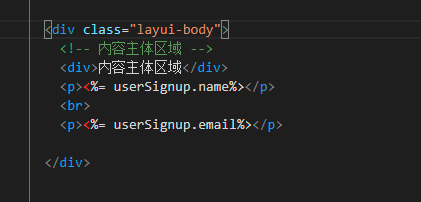
The only thing we need to change is



To

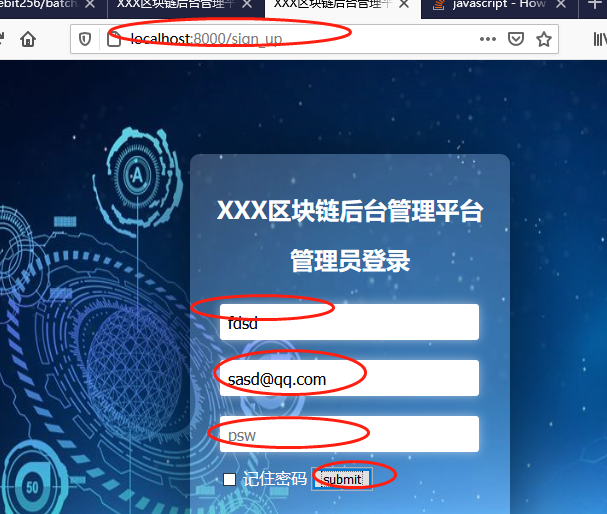


5 add the username to the index

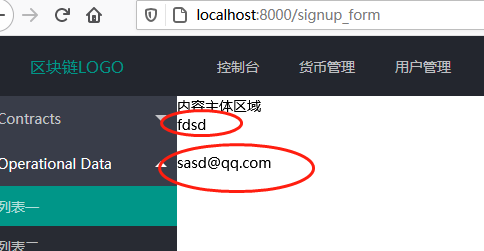


6 Test

Run the application, after type into the name email and psw

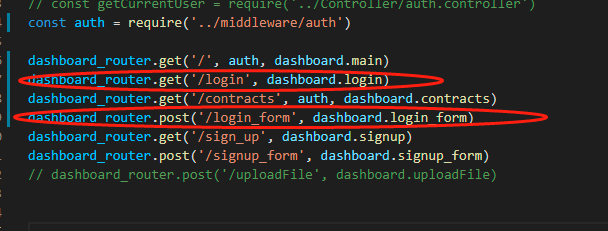


You will be able to see

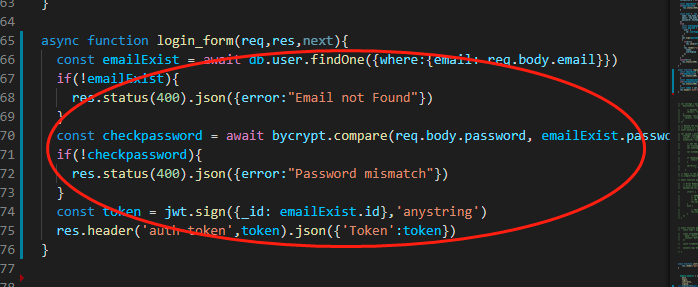


## 3.2 login

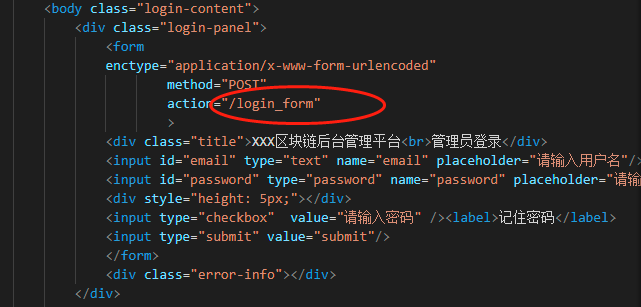
1 add router



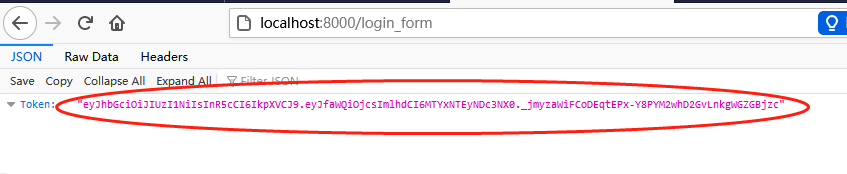
2 add controller and copy the login logic from api’s controller



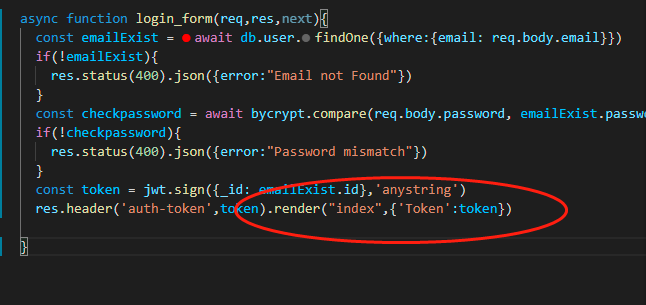
3 add view



4 verify



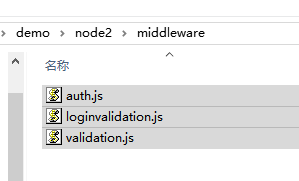
5 change json return to page render



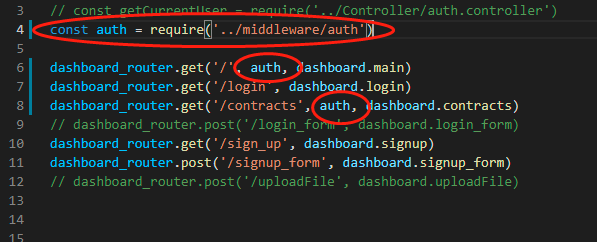
Why render? Not redirect

# 4 add auth middleware to all routes except login page

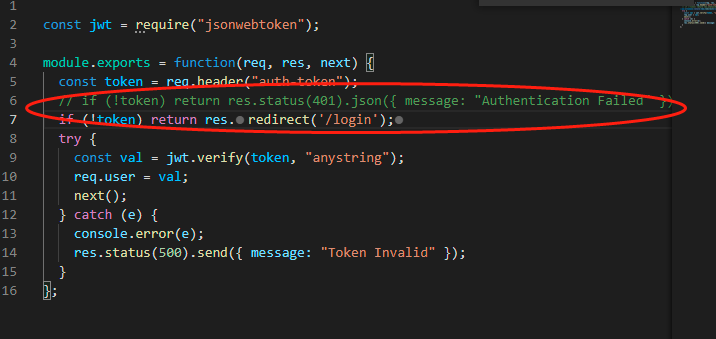
1. Create a folder called middleware and copy the middleware from the project 2



1. Add middleware



1. Redirect unauthorized user to login page: change auth.js from api redirect to page redirect

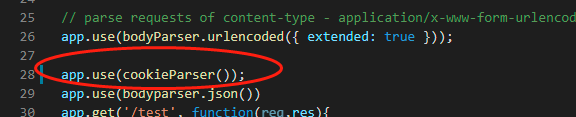


1. Verify open 127.0.0.1:8000, will be redirected to 127.0.0.1:8000/login

# 5 login use cookie

Npm i cookie-parser

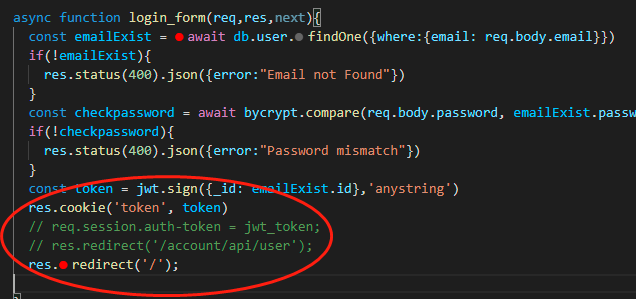
1. Add cookie



2 use cookie to store token



3 add cookie to the controller



4 verify

After the first login, you don’t need to psw to login, cookie will keep your JWT

# 6 fileupload