**EXPRESS JS**

# 1) Cài đặt Express JS trong Project

npm init

npm install express --save

# 2) Khởi tạo Express

const express = require('express')

const app = express()

const port = 3000

app.listen(port, function() {

console.log(`Example app listening on port + port)

})

# 3) Route

* app.get('/path', function(req, res){

res.send(‘<h1>Hello world</h1>’) ;

})

*Trong đó :*

/path: Đường dẫn chỉ định

req: Chứa tất cả thông số mà khách hàng truyền lên

res: Chứa tất cả thông số mà server tải về

send: Method của res

=> Method **get**: Khi có 1 get request gửi lên /path thì function callback với 2 tham số req,res sẽ được gọi và method res.send sẽ trả về 1 cái gì đó

# 4) Template engine

## a) Setup:

* npm install pug --save
* ***Cài đặt template engine:*** app.set('view engine', 'pug')

pug: Template engine sử dụng

* ***Cài đặt thư mục gốc:*** app.set('views', './views')

./views: Thư mục gốc chỉ định

## b) Use

* app.get('/path', function (req, res) {

res.render('index', { title: 'Hey', message: 'Hello there!' })

})

*Trong đó :*

/path: Đường dẫn chỉ định

res.render: Method này sẽ compile pug sang html

index: Tên của file pug (Tham số thứ nhất)

{title: 'Hey', message: 'Hello there!'}: Là một object truyền. Chúng ta có thể lấy trực tiếp giá trị từ biến này ở trong index (Tham số thứ hai không bắt buộc)

NOTE: method render nhận vào 2 tham số

# 5) Query parameters



Đoạn mã trên nhằm mục đích tìm kiếm trong danh sách những thứ liên quan đến “my tam”

## a) Use

* req.query.q: để get giá trị của q
* form(action="/users/search" method="GET")

input (type="text" name="q")

button

Đoạn code trên nhằm mục đích gửi lên đường dẫn giá trị của ***q*** người dùng nhập ở trong input

# 6) Post method

## a) Use

* req.body: Trả về 1 cặp key value trong dữ liệu được gửi đi. Mặc định là undefined, ta có thể dùng body-parsing middleware như **body-parse** để phân tích nó ra các cặp key value nằm trong object

Cài đặt **body-parse:**

- npm install body-parse --save

- app.use(bodyParser.json());

- app.use(bodyParser.urlencoded({ extended: true }))

* Ví dụ:

app.post('/path', function (req, res) {

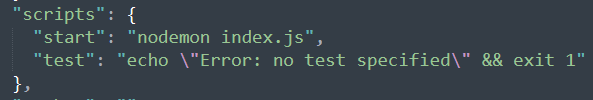
console.log(req.body)

})

# 7) Nodemon

## a) Install

* npm install --save-dev nodemon
* Thêm vào file package.jsom như sau:



## b) Use

Như vậy khi start server ta chỉ cần chạy lệnh nodemon index.js thì terminal sẽ tự động reload lại khi có thay đổi trên file index

# 8) Lowdb

## a) Install

* npm install lowdb
* const low = require('lowdb')
* const FileSync = require('lowdb/adapters/FileSync')
* const adapter = new FileSync('db.json')
* const db = low(adapter)

## b) Use

* db.defaults({ db: []})

.write()

db: [] là giá trị của 1 array mặc định mà sau đó ta sẽ truyền dữ liệu trong file db.json vào

NOTE: Nếu project có dùng nodemon thì phải tạo 1 file nodemon.json và ignore file db.json

* db.get('db').value(): Trả về giá trị trong file db.json

value(): Method để lấy giá trị bên trong file json

db: Tên của file database.json

db: Tên của giá trị mặc định ta đặt lúc đầu

* Ghi dữ liệu vào file database:

db.get('db ').push({name: John}).write()

# 9) View user

## a) Use

* Route path: /users/:userId

Request URL: http://localhost:3000/users/34/

req.params: { "userId": "34"}

* router.get('path/:id', function(req,res){

var id = req.params.id;

var user = db.get('users').find({id: id}).value()

res.render('users/view', {

user: user

})

})

*Trong đó:*

path/:id: Cứ khi nào có 1 get request gửi đến /path/một id nào đó thì chạy callback

req.params.id: Một id nào đó được lưu vào biến này

db.get('users').find({id: id}).value(): Method của lowdb sẽ tìm kiếm và lấy giá trị id trùng với id được

## b) Create id using shortId

* Install: npm install shortid --save
* Use:

Khai báo thư viện: var ids **=** require('short-id');

Sử dụng: ids.generate()

# 10) Express Router

- Dùng để chia nhỏ chương trình thành những folder riêng biệt chứa các file có chức năng giống nhau

## a) Use

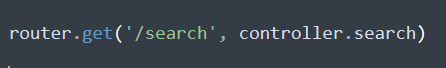
* Tạo 1 folder router bên trong chứa file user.router.js
* Khai báo router: var router = express.Router()
* Để những thứ liên quan vào trong 1 file js (VD: app.get(/user/…) thì cho những gì liên quan đến user vào 1 file user.route.js
* Thay thế app. = router. , đồng thời xóa /user
* Sau đó export file js ra bằng lệnh module.exports = user;
* Trong file main.js require file moudle user bằng lệnh var userRoutes = require('./routes/user.route')
* Sau đó sửa callback đằng sau của app thành app.use('/users', userRoutes);

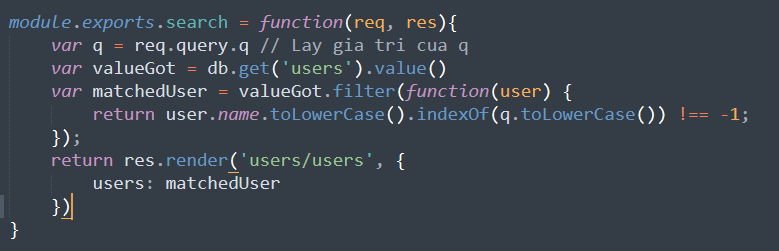
NOTE: Khi chia nhỏ file ra mà file có có yêu cầu module cần thiết thì phải require module đó vào

# 11) Controller

- Những thứ liên quan đến logic xử lý sẽ được tách riêng là 1 file controller

## a) Use

* Tạo 1 folder controllers bên trong chứa file user.controller.js
* Trong file user.router.js require module controller vào
* Trong file user.router.js sửa callback thành
* Trong file user.controller.js viết



12) Static file

a) Use

* app.use(express.static('public'))

public: Thư mục public này chứa các file tĩnh như css, js, images …

* Sau khi đã khai báo ta chỉ cần dẫn link /css /js /images là được