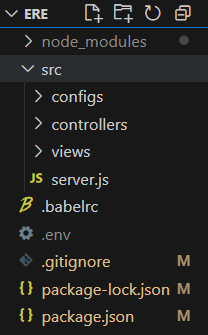
**ExpressJS**

npm i express ejs dotenv

Cấu trúc express src **SSR** với **ejs**

* **src**
* routes: khai báo route + method
* controllers: xử lý request, response của route
* service: logic của controller
* views: file ejs
* public: css,html
* configs: config view engine, …



* babel dùng để dịch code của nodejs
* gitignore chặn folder, file đẩy lên git
* env là tham số môi trường

.gitignore:

.env

node\_modules

node\_modules/\*

.babelrc

{

    "presets": [

        [

            "@babel/preset-env",

        ]

    ]

}

- Dùng babel để dịch code

npm install --save-dev @babel/core @babel/cli @babel/preset-env @babel/node

- Dùng nodemon để chạy lại server

npm i nodemon

**View engine (config)**

import express from 'express'

const useViewEngine = (app) => {

    app.use(express.static('./src/public')) // Vị trí cho các file html css

    app.set('view engine', 'ejs') // Sử dụng ejs

    app.set('views', './src/views') // Vị trí folder chứa file ejs

}

**Router (Khai báo route trong web.js)**

import { express, Router } from 'express'

const router = Router()

const appRoutes = (app) => {

    //Khai báo các route: "/", "/hello"

    router.get('/', (req, res) => {

        res.send('Get home page')

    })

    router.get('/hello', (req, res) => {

        res.send('Get hello message')

    })

    return app.use('/', router) // Root route, truyền route vào app

}

export default appRoutes

*Dùng router để* ***tách code*** *ra, router dùng để* ***tổng hợp các url và method*** *và sau đó* ***truyền vào app*** *express*

const router = Router()

**Chạy app bằng nodemon (Chỉ nodemon dùng đc babel)**

  "scripts": {

    "dev": "nodemon --exec babel-node src/server.js"

  },

**Chạy Server**

import express from 'express'

import useViewEngine from './configs/viewEngine'

import useRoutes from './routes/web'

require('dotenv') // get data from env

const app = express()

const PORT = process.env.PORT || 8080

useViewEngine(app)

useRoutes(app)

// app.listen = run app

app.listen(PORT, () => {

    console.log(`Example app listening on PORT ${PORT}`)

})

})

**Controller**

// Xử lý req,res tại controller

const handleHomePage = (req, res) => {

    return res.send('Get home page')

}

const handleHello = (req, res) => {

    return res.send('Get hello page')

}

export { handleHomePage, handleHello }

***Connect to SqlDB***

* npm i mysql2

***Encrypt password***

* npm i bcryptjs

***Get body request as json {}***

* npm i body-parser

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

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

**=> const username = req.body.username**

const handleHomePage = (req, res) => {

    const username = req.body.username

    return res.render('index.ejs', { username })

}

**Service**

import mysql from 'mysql2/promise';

const getAllUsers = async () => {

    const connection = await mysql.createConnection({

        host: 'localhost',

        user: 'root',

        database: 'firstdb',

    });

    try {

        const [results, fields] = await connection.query(

            'SELECT \* FROM `user` '

        );

        return results; // results contains rows returned by server

    } catch (err) {

        console.log(err);

    }

}

export { getAllUsers }

**Sequelize ORM**

npm install --save sequelize

**Connect** tới firstdb

import { Sequelize } from "sequelize";

const connectDB = async () => {

    const sequelize = new Sequelize('firstdb', 'root', '', {

        host: 'localhost',

        dialect: "mysql",

        define: {

            freezeTableName: true, // Không thêm s vào table

        },

    });

    try {

        await sequelize.authenticate();

        console.log('Connection has been established successfully.');

    } catch (error) {

        console.error('Unable to connect to the database:', error);

    }

}

export default connectDB

**Sequelize CLI (Migrations)**

[**https://sequelize.org/docs/v6/other-topics/migrations/**](https://sequelize.org/docs/v6/other-topics/migrations/)

Folder config -> config.json

{

  "development": {

    "username": "root",

    "password": null,

    "database": "firstdb",

    "host": "127.0.0.1",

    "dialect": "mysql"

  },

...

}

.sequelizerc cấu hình path để dùng CLI

const path = require('path');

module.exports = {

    'config': path.resolve('./src/config', 'config.json'),

    'migrations-path': path.resolve('./src/migrations'),

    'models-path': path.resolve('./src/models'),

    'seeders-path': path.resolve('./src/seeders'),

};

• config: config file

• models: chứa các model

• migrations: Tạo bảng tương ứng với model

•seeders: Fake data cho bảng

* *CLI* ***tự động thêm folder***

npx sequelize-cli init

* *CLI tự động tạo model*

npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string

* *CLI tạo table trên database*

npx sequelize-cli db:migrate

* *CLI tạo fake data cho table*

npx sequelize-cli seed:generate --name demo-user

**INSERT sequelizeCLI**

import db from '../models/index'

const addUser = async (userName) => {

    try {

        await db.User.create({

            userName: 'asdjahd',

            age: 19

        })

    } catch (error) {

        console.log(error)

    }

}

**Get one: get({ plain: true })**

    const data = await db.User.findOne({ where: { id: 1 } });

    console.log(data.get({ plain: true }))