|  |
| --- |
| **EPRESS JS** |
| **1. Khái niệm** |
| - Epress JS cung cấp nhưng công cụ cần thiết để xây dựng ứng dụng  - Epress hướng tới sự linh động , chỉ thêm vào những gì cần thiết  - Epress cung cấp một số tính năng quan trọng như Middleware ( hàm xử lý trung gian) , đơn giản hoá khả năng điều hướng của ứng dụng.  - Ngoài ra Epress tích hợp rất nhiều Template Engine giúp dễ tương tác với HTML |
| **2. Cài đặt** |
| **npm i express** *// Cài đặt trực tiếp vào dependency*  **Sử dụng express trong NodeJS:**  **const express = require(‘express’)**  **const app = express();** |
| **3. Tạo web server với Express** |
| const **express** = require('express');  // Hàm khởi tạo của Epress , tạo ra một đối tượng  const **app** = express();    // Khai báo Router cho ứng dụng và gọi đến Hàm xử lý  app.get('/', function(req, res){     res.send("Hello world!");  });    // Chỉ định ứng dụng lắng nghe ở cổng nào  app.listen(3000);  --> Mặc định sẽ là: <http://localhost>: 3000 ; nhưng ta cũng có thể thay đổi  app.listen(3000 , “newlocalhost”);  🡺 http://newlocalhost: 3000 |
| **4. Route trong Express** |
| **Tạo bộ định tuyến:**  1. Tạo file js chứa bộ định tuyến: NewRouter.js  **const express = require('express');**  **const router = express.Router();**    **router.get('/', function(req, res){**  **res.send('Ban da truy cap dia chi /user bang phuong thuc GET');**  **});**  **router.post('/', function(req, res){**  **res.send('Ban da truy cap dia chi /user bang phuong thuc POST');**  **});**    **// Xuất bộ định tuyến này để có thể sử dụng ở file khác**  **module.exports = router;**  2. Tại file index.js ta sử dụng bộ định tuyến NewRouter.js như sau    const **NewsRouter** = require('./ NewRouter);  app.**use**('/user', **NewsRouter**);    app.listen(3000); |
| **5. URL trong Express** |
| **1. STATIC URL : đường dẫn tĩnh**  - Đường dẫn tĩnh là đường dẫn cố định,  vd: https://example.com/home  **2. Dynamic URL : đường dẫn động**  - Đường dẫn động là đường dẫn điều hướng được tới nhiều nơi hơn vd:  https://example.com/posts/1  https://example.com/posts/2  <https://example.com/posts/3>  **3. Router parametes: Các tham số trong đường dẫn:**  Route path: /users/:**userId**/books/:**bookId**  Request URL: http://localhost:3000/users/**34**/books/**8989**  req.params: { "userId": "34", "bookId": "8989" } |
| **6. Hàm xử lý sự kiện : function( req , res ) trong Express** |
| **1. REQ - Request Object**  - Request Object - kiểm tra các yêu cầu mà Client gửi lên Server   |  |  | | --- | --- | | **Prop** | **Description** | | **req.params()** | Trả về một Object chứa các thuộc tính cho việc định danh Router ( **Router parameter** ) | | **req.query** | Trả về một Object chứa các **Query Parameter** | | **req.body()** | Trả về một Object chứa KEY\_VALUE gửi bằng phương thức POST |   **2. RES - Response Object**  - Response Object - Phản hồi các yêu cầu HTTP   |  |  | | --- | --- | | **Method** | **Description** | | **res.send( )** | Gửi lại các loại dữ liệu như **HTML** hoặc đoạn **text** thông thường | | **res.json( )** | Phản hồi loại **json** của dữ liệu phản hồi | | **res.status( )** | Chỉ định **mã phản hồi**. vd: 404 - forder not found | | **res.redirect( )** | Chuyến hướng đến route khác | | **res.render( )** | Phản hồi lại nội dung của file HTML chỉ định và có thể truyền tham số là Object chứa các giá trị muốn truyền vào file HTML trước khi gửi về Client | | **res.end( )** | Kết thúc phản hồi đến Client | |
| **7. Middleware trong Epress** |
| **Middleware : cầu nối kết nối các phần với nhau**  - Middleware giúp chặn các request xấu,  thực thi xử lý chuyển đổi một số request trước khi  các yêu cầu hoàn tất để response  **Middleware Function : function( req , res , next )**  - Middleware Function luôn được thêm vào theo  thứ tự từ trên xuống thực hiện đồng bộ với nhau.  - Các middleware function có các chức năng sau:   * Thay đổi các đối tượng **req** và **res** * Kết thúc vòng đời của request ( nếu muốn chặn Request xấu ) * Gọi các middleware trong stack ( sắp xếp thứ tự )   Hàm **next( )** dùng để chuyển sang middleware tiếp theo  vd:  var checkRequest = (req, res, next) => {      console.log('Middleware chạy ở route có url ' + req.url + ' và method là ' + req.method )      if (req.url === '/block') {          res.send('Bạn không có quyền truy cập !')      }else{          next()      }  }  app.use(checkRequest ); // kiểm tra nếu url path = “block” sẽ ngăn không có thực hiện các request tiếp theo.  Các loại Middleware trong Express :  **1. Application-level Middleware :**  **Các middleware trực tiếp tại ứng dụng**  app.use(function (req, res, next) {    console.log('Receive all request')    next()  })  **hoặc :**  app.get('/user/:id', function (req, res, next) {    console.log('ID:', req.params.id)    next()  }, function (req, res, next) {    res.send('User Info')  })  **2. Router-level Middleware :**  **Các middleware tại phần router của ứng dụng:**  router.use(function (req, res, next) {    console.log('middleware run')    next()  })  **3. Error-handling middleware:**  **Các middleware bắt lỗi khi có lỗi xảy ra:**  - Truyền đối số err vào hàm Function middleware tại vị trí đối số thứ nhất;  app.use(function (err ,req, res, next) {    console(err.stack)    next()  })  **4. Third-party middleware:**  **Xử dụng bên thứ 3 để làm middle-ware**   * **body-parser : Module giúp lấy được dữ liệu nhập vào method = POST**   Cài đặt : npm i body-parser  Xuất : const bodyParser = require(‘body-parser’);  Sử dụng: app.use( bodyParser.urlencoded( {extended : true} );   * **morgan : Module theo dõi tất cả HTTP request**   Cài đặt : npm i body-parser --save-dev  Xuất : const morgan = require(‘morgan);  Sử dụng: app.use( morgan(‘combined’) );   * **express-handlebars : Hỗ trợ template engine**   Cài đặt : npm i express-handlebars  Xuất : const handlebar = require(‘express-handlebars’);  Sử dụng: app.engine( ‘hbs’,  handlebars({  extname: ‘.hbs’,  helper: {}  } );  app.set( ‘view engine’ , ‘hbs’ );  app.set(‘views’, “đường dẫn tới nới để file chứa các file hbs”   * **x** * **x** * **x** |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |