**Bài tập tổng hợp Express, Routing và Middleware**

## 1. Mục tiêu

* Hiểu được các kiến thức cơ bản về thư viện Express.js
* Hiểu và vận dụng được cách cài đặt môi trường phát triển Express.js
* Hiểu được khái niệm và nắm được cách routing nâng cao trong Express.js.
* Hiểu và nắm được cơ chế hoạt động của Middleware trong Express.js.
* Hiểu và nắm được cơ chế serving static file trong Express.js.
* Hiểu và nắm sự khác biệt giữa SSR (Server-side rendering) website vs CSR (Client-side rendering) website
* Hiểu được cách tạo một CSR dynamic website.

## 2. Mô tả

* Express.js là một thư viện được build trên nền Node.js – Một môi trường giúp Javascript có thể chạy được trên môi trường phát triển server và giúp code ra được những SSR (server-side rendering) dynamic website
* Ngoài ra, Express.js cũng là một công cụ rất mạnh, giúp chúng ta có thể build được những API (Application Programming Interface) - Tổ hợp các endpoint trả về hoặc xử lý dữ liệu dạng JSON để build được những CSL (client-side rendering) dynamic website
* Đối với những server-side rendering website **(1)**

→ Tổ hợp những endpoint sẽ giúp người dùng website (client) điều hướng giữa các trang (pages) với nhau và ở mỗi endpoint sẽ render ra trang HTML của riêng endpoint đó với data được gắn trực tiếp vào HTML thông qua các template engines như ejs, pug, handlebars, …

* Đối với những client-side rendering website, tổ hợp những endpoint sẽ được chia ra làm 2 loại:
  + Endpoint gửi dữ liệu về client dưới dạng json, (2)

1. Việc routing giữa các trang (pages) và render ra những trang HTML tại các endpoint của người dùng website, sẽ do một thư viện frontend bên thứ 3 xử lý.

2. Công việc duy nhất của server là tổng hợp những endpoint gửi và nhận dữ liệu json đó lại thành một bộ API document (Application Programming Interface) để các thư viện bên thứ 3 có thể gọi dữ liệu và render ra trang HTML mỗi khi cần

* + Endpoint gửi dữ liệu về client dưới dạng json, NHƯNG **(3)**

1. Việc routing giữa các trang (pages) và render ra những trang HTML tại các endpoint của người dùng website vẫn sẽ do server xử lý (Thế mới hay) =))))

2. Server sẽ có nhiệm vụ tổng hợp lại những endpoint gửi và nhận những dữ liệu json đó lại thành một bộ API document (Application Programming Interface)

3. Vì các endpoint gửi dữ liệu json về client đã được định nghĩa và sẽ chỉ giữ nhiệm vụ gửi và nhận data dưới dạng json từ server về client ⇒

4. Sẽ phải có những route (endpoint) xử lý việc render ra các trang HTML để người dùng website (client) có thể điều hướng và view nội dung

5. Các file HTML tĩnh này sẽ sử dụng Javascript thuần để gọi (fetch) đến các API được server định nghĩa, lấy dữ liệu và render.

**TÓM LẠI**

* Trước khi xây dựng bất kỳ một website nào, phải xác định rõ kiểu website (SSR, CSL) muốn xây dựng
* Sau khi xác định được dạng website
  + Đối với SSR → Không cần build API
  + Đối với CSL → Build API.
* Khi đã chọn các website dạng CSL → Chúng ta sẽ phải follow một quy chuẩn để viết API. Có rất nhiều dạng quy chuẩn viết API, và một dạng nổi tiếng nhất được các developers tin dùng đó là chuẩn RESTFUL (RESTFUL API)
* Trong Session này, chúng ta sẽ học cách build các website dạng (3).
* Các website dạng (2) sẽ được học khi chúng ta có kiến thức về các thư viện frontend bên thứ 3 như ReactJS, VueJS hoặc AngularJS
* Các website dạng (1) sẽ được học khi chúng ta có kiến thức về các template engines như ejs, pug…
* Trong Session này chúng ta sẽ build một con app khá cool có tên là Ask Community với kiểu website dạng (3) được note lại ở trên
* Tiến hành tải thư mục dev-data/ có chứa 2 file questions.json và backup-questions.json và cho vào thư mục “ask-community-project”

<https://drive.google.com/drive/folders/1vnsu_8nzQXgS3uahTgi6wfT-L8BnGpdM?usp=sharing>

Trong file questions.json là một mảng chứa thông tin về các questions ngẫu nhiên

Trong exercise 03 này chúng ta sẽ build một api nhỏ để thực hành các thao tác gửi

GET (Read)/ POST (Create)/ PUT (update)/ DELETE (delete) request đến endpoint và xử lý chúng

NOTE:

* Sử dụng POSTMAN để test api !!!
* Khi đọc file dạng “.json", data lấy về được phải sử dụng JSON.parse(data) để convert về dạng Javascript đọc được
* Khi ghi dữ liệu vào file “.json", trước khi được ghi vào phải sử dụng JSON.stringify(data) để convert data về dạng json.

Bước 1:

* Tạo GET request với đường dẫn “/api/v1/questions”
* Gõ GET request đến “/api/v1/questions” trong POSTMAN
* Tiến hành đọc file questions.json thông qua câu lệnh readFile()
* Lấy ra một mảng questions và trả về cho người dùng dưới dạng json thông qua câu lệnh res.status(code).json();

Bước 2:

* Tạo GET request với đường dẫn “/api/v1/questions/:id”
* Gõ GET request đến “/api/v1/questions/${id}” trong POSTMAN
* Tiến hành lấy ra id dựa vào req.params.id
* Tiến hành đọc file questions.json thông qua câu lệnh readFile()
* Tiến hành tìm kiếm question trong mảng questions thông qua id
* Trả về cho người dùng dưới dạng json question vừa tìm kiếm được thông qua câu lệnh res.status(code).json()

Bước 3:

* Tạo POST request với đường dẫn “/api/v1/questions”
* Gõ POST request đến “/api/v1/questions” trong POSTMAN
* Nhập đầy đủ các trường thông tin vào trong phần body/form-data
* Tiến hành cài đặt module body-parser từ npm (npm i body-parser)
* Require body-parser vào file app.js và sử dụng thông qua app.use()
* Tiến hành lấy data người dùng gửi từ POST request thông qua req.body
* Tiến hành đọc file questions.json thông qua câu lệnh readFile()
* Tiến hành tìm kiếm user theo trường content
  + Nếu không tìm thấy question
    - Tiến hành ghi đè vào file questions.json thêm mới question vừa lấy được tại req.body
    - → trả về cho người dùng dưới dạng json message {message: “Create successfully” } → res.status(code).json(message)
  + Nếu tìm thấy question
    - → trả về cho người dùng dưới dạng json message {message: “Question already exists” } → res.status(code).json(message)

Bước 4:

* Tạo PUT request với đường dẫn “/api/v1/questions/:id”
* Gõ PUT request đến “/api/v1/questions/${id}” trong POSTMAN
* Nhập đầy đủ các trường thông tin muốn update vào trong phần body/x-www-form-urlencoded
* Tiến hành cài đặt module body-parser từ npm (npm i body-parser)
* Require body-parser vào file app.js lần đầu tiên và sử dụng thông qua app.use()
* Tiến hành lấy data người dùng gửi từ POST request thông qua req.body
* Tiến hành lấy ra id dựa vào req.params.id
* Tiến hành đọc file questions.json thông qua câu lệnh readFile()
* Tiến hành tìm kiếm question trong mảng questions thông qua id
  + Nếu không tìm thấy question → trả về cho người dùng dưới dạng json message {message: “Question not found” } → res.status(code).json(message)
  + Nếu tìm thấy question
    - Tiến hành ghi đè vào file questions.json thêm mới các trường thông tin trong vừa lấy được tại req.body
    - Trả về cho người dùng dưới dạng json message {message: “Update successfully” } → res.status(code).json(message)

Bước 5:

* Tạo DELETE request với đường dẫn “/api/v1/questions/:id”
* Gõ DELETE request đến “/api/v1/questions/${id}” trong POSTMAN
* Tiến hành lấy ra id dựa vào req.params.id
* Tiến hành đọc file questions.json thông qua câu lệnh readFile()
* Tiến hành tìm kiếm question trong mảng questions thông qua id
  + Nếu không tìm thấy question → trả về cho người dùng dưới dạng json message {message: “Question not found” } → res.status(code).json(message)
  + Nếu tìm thấy question
    - Tiến hành xóa question vừa tìm thấy trong mảng questions và ghi đè vào file questions.json
    - Trả về cho người dùng dưới dạng json message {message: “Delete successfully” } → res.status(code).json(message)

## 3. Đánh giá

Để hoàn thành bài tập, học viên cần:

* Đưa mã nguồn lên GitHub.
* Dán link của repository lên phần nộp bài trên hệ thống.