# Bài giảng: Routing Toàn Stack với React Router và Express

## 1. Giới thiệu

Routing (định tuyến) là một phần quan trọng trong ứng dụng web fullstack. Ở phía client (React), ta dùng React Router để điều hướng giữa các trang mà không cần tải lại toàn bộ trang. Ở phía server (Express), routing giúp xử lý request từ client và trả về response hoặc dữ liệu API.

## 2. Routing phía Client với React Router

Cài đặt React Router:

npm install react-router-dom

Ví dụ App.jsx:

import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";  
import Home from "./pages/Home";  
import About from "./pages/About";  
import Contact from "./pages/Contact";  
  
export default function App() {  
 return (  
 <Router>  
 <nav>  
 <Link to="/">Home</Link> |   
 <Link to="/about">About</Link> |   
 <Link to="/contact">Contact</Link>  
 </nav>  
 <Routes>  
 <Route path="/" element={<Home />} />  
 <Route path="/about" element={<About />} />  
 <Route path="/contact" element={<Contact />} />  
 </Routes>  
 </Router>  
 );  
}

• BrowserRouter: cung cấp ngữ cảnh routing cho toàn ứng dụng.

• Routes/Route: định nghĩa đường dẫn và component hiển thị.

• Link: thay thế thẻ <a>, điều hướng mà không reload trang.

## 3. Nested Routing

Ví dụ: Trang About có sub-route Team và History:

<Route path="/about" element={<About />}>  
 <Route path="team" element={<Team />} />  
 <Route path="history" element={<History />} />  
</Route>

## 4. Routing phía Server với Express

Ví dụ server.js:

const express = require("express");  
const app = express();  
const PORT = 5000;  
  
app.get("/", (req, res) => {  
 res.send("Trang chủ từ Express");  
});  
  
app.get("/api/users", (req, res) => {  
 res.json([{ id: 1, name: "Alice" }, { id: 2, name: "Bob" }]);  
});  
  
app.listen(PORT, () => console.log(`Server chạy tại http://localhost:${PORT}`));

• app.get/post/put/delete: định nghĩa route cho các phương thức HTTP.

• res.send(): trả về HTML/text.

• res.json(): trả về dữ liệu JSON (thường dùng cho API).

## 5. Kết hợp React Router và Express

Quy trình:

1. 1. Express chạy backend, cung cấp API tại /api/\*.
2. 2. React Router quản lý giao diện SPA phía client.
3. 3. React gọi API từ Express bằng fetch/axios.

Ví dụ gọi API trong React:

import { useEffect, useState } from "react";  
  
export default function Users() {  
 const [users, setUsers] = useState([]);  
  
 useEffect(() => {  
 fetch("/api/users")  
 .then(res => res.json())  
 .then(data => setUsers(data));  
 }, []);  
  
 return (  
 <div>  
 <h2>Danh sách người dùng</h2>  
 <ul>  
 {users.map(u => <li key={u.id}>{u.name}</li>)}  
 </ul>  
 </div>  
 );  
}

## 6. Bài tập thực hành

• Tạo ứng dụng React với 3 trang: Home, Products, Contact, điều hướng bằng React Router.

• Tạo Express server với route /api/products trả về danh sách sản phẩm (JSON).

• Gọi API từ React và hiển thị danh sách sản phẩm ở trang Products.

## 7. Câu hỏi trắc nghiệm

Câu 1: Trong React Router, component nào được dùng để định nghĩa đường dẫn?  
A. <Link> B. <Routes> C. <Route> D. <Switch>  
Đáp án đúng: C — Giải thích: <Route> khai báo đường dẫn và component hiển thị.

Câu 2: Trong Express, phương thức nào dùng để trả về dữ liệu JSON?  
A. res.send() B. res.json() C. res.end() D. res.write()  
Đáp án đúng: B — Giải thích: res.json() tự động set Content-Type là application/json.

Câu 3: Khi kết hợp React Router và Express, vai trò của Express thường là gì?  
A. Điều hướng trang client B. Cung cấp API và xử lý request server  
C. Render JSX D. Tạo component React  
Đáp án đúng: B — Giải thích: Express là backend, cung cấp API/route server cho React tiêu thụ.