Chapter 04: React Router - Xây dựng ứng dụng đa trang (SPA)

A. Mục tiêu

Sau bài học này, học viên sẽ:

- Hiểu khái niệm **Single Page Application (SPA)** và vai trò của **client-side routing**.
- Biết cách cài đặt và cấu hình thư viện `react-router-dom`.
- Xây dựng được các **route tĩnh** và **route động**.
- Thành thạo sử dụng các component chính: `BrowserRouter`, `Routes`, `Route`, `Link`, `NavLink`.
- Sử dụng hook `useParams` để truy xuất tham số từ URL động.

B. Nội dung lý thuyết

1. Single Page Application (SPA) là gì?

SPA là một ứng dụng web chỉ tải một trang HTML duy nhất từ server. Khi người dùng điều hướng, JavaScript động cập nhật nội dung giao diện mà không cần tải lại toàn bộ trang.

Ưu điểm

- Trải nghiệm người dùng mượt mà, tương tự ứng dụng desktop.
- Tốc độ điều hướng nhanh sau lần tải ban đầu.
- Giảm tải cho server vì chỉ cần gửi dữ liệu JSON thay vì HTML hoàn chỉnh.

Nhược điểm

- Thời gian tải ban đầu có thể lâu hơn do tải toàn bộ JavaScript.
- Yêu cầu xử lý **client-side routing** để đồng bộ URL với giao diện.

- Có thể gặp vấn đề về **SEO** nếu không cấu hình server-side rendering hoặc pre-rendering.

2. Giới thiệu react-router-dom

`react-router-dom` là thư viện phổ biến nhất để triển khai routing trong ứng dụng React. Nó cung cấp các công cụ để quản lý điều hướng và ánh xạ URL với các component.

Cài đặt

Chạy lệnh sau để cài đặt:

```
npm install react-router-dom
```

Các Component cốt lõi

- `<BrowserRouter>` : Bao bọc toàn bộ ứng dụng, sử dụng **HTML5 History API** để đồng bộ URL với giao diện. Thường đặt trong `index.js` hoặc `App.js` .
- `<Routes>` : Chứa các `<Route>` con, chọn route đầu tiên khớp với URL hiện tại để render.
- `<Route>` : Định nghĩa ánh xạ giữa **path** (đường dẫn) và **element** (component giao diện).

```
<Route path="/about" element={<AboutPage />} />
```

- `<Link>` : Tạo liên kết điều hướng, thay thế thẻ `<a>` để tránh tải lại trang.

```
<Link to="/about">Về chúng tôi</Link>
```

- `<NavLink>` : Biến thể của `<Link>` , tự động thêm class `active` khi URL khớp với `to` , hỗ trợ style cho liên kết đang được chọn.

```
<NavLink to="/home" className={({ isActive }) => isActive ? 'active-link' :
```

3. Route động (Dynamic Routes)

Route động cho phép xử lý các URL có tham số, ví dụ: `/products/:productId`. Dấu `:` trong `path` biểu thị một tham số động.

Ví dụ:

```
<Route path="/products/:productId" element={<ProductDetailPage />} />
```

4. Hooks của React Router

- `useParams()`: Trả về object chứa các tham số từ URL. Ví dụ, với `/products/123`, `useParams()` trả về `{ productId: '123' }`.

```
import { useParams } from 'react-router-dom';
function ProductDetailPage() {
  const { productId } = useParams();
  return <h1>Sản phẩm {productId}</h1>;
}
```

- `useNavigate()`: Cho phép điều hướng lập trình (programmatic navigation), ví dụ: chuyển hướng sau khi submit form.

```
import { useNavigate } from 'react-router-dom';
function LoginButton() {
  const navigate = useNavigate();
```

```
  return <button onClick={() => navigate('/home')}>Đăng nhập</button>;
}
```

- `useLocation()` : Cung cấp thông tin về URL hiện tại (pathname, search, hash).

```
import { useLocation } from 'react-router-dom';
function CurrentPage() {
  const location = useLocation();
  return <p>URL hiện tại: {location.pathname}</p>;
}
```

C. Bài tập thực hành

Bài 1: Chuyển đổi Todo List thành SPA

- **Yêu cầu**: Chuyển ứng dụng Todo List từ bài 2 thành SPA với 3 trang:
  - `/` : Hiển thị tất cả công việc.
  - `/active` : Hiển thị công việc chưa hoàn thành.
  - `/completed` : Hiển thị công việc đã hoàn thành.
  - Sử dụng `<NavLink>` để tạo menu điều hướng với style cho tab active.
- **Hướng dẫn**:

1. Cài đặt `react-router-dom` :

```
npm install react-router-dom
```

2. Cập nhật `src/App.jsx` :

```jsx
import { useState } from 'react';
import { BrowserRouter, Routes, Route, NavLink } from 'react-router-dom';
import TodoForm from './components/TodoForm';
import TodoList from './components/TodoList';

function App() {
  const [todos, setTodos] = useState([]);

  function addTodo(text) {
    setTodos([...todos, { id: Date.now(), text, completed: false }]);
  }

  function deleteTodo(id) {
    setTodos(todos.filter(todo => todo.id !== id));
  }

  function toggleComplete(id) {
    setTodos(
      todos.map(todo =>
        todo.id === id ? { ...todo, completed: !todo.completed } : todo
      )
    );
  }

  return (
    <BrowserRouter>
      <div>
        <h1>Todo List</h1>
        <nav>
          <NavLink to="/" className={({ isActive }) => isActive ? 'active' :
          <NavLink to="/active" className={({ isActive }) => isActive ? 'act
          <NavLink to="/completed" className={({ isActive }) => isActive ? '
        </nav>
        <TodoForm onAdd={addTodo} />
        <Routes>
          <Route path="/" element={<TodoList todos={todos} onDelete={deleteT
          <Route path="/active" element={<TodoList todos={todos.filter(todo
          <Route path="/completed" element={<TodoList todos={todos.filter(to
        </Routes>
      </div>
    </BrowserRouter>
  );
```

```
}
export default App;
```

3. Thêm style trong `src/index.css`:

```
.active {
  font-weight: bold;
  color: blue;
}
nav a {
  margin-right: 10px;
  text-decoration: none;
}
```

Bài 2: Xây dựng trang Blog đơn giản

- **Yêu cầu**: Tạo một mảng bài blog giả. Trang `/blog` hiển thị danh sách tiêu đề, mỗi tiêu đề là `<Link>` dẫn đến `/blog/:postId`. Trang chi tiết sử dụng `useParams` để hiển thị bài blog.
- **Hướng dẫn**:

1. Tạo `src/components/BlogList.js`:

```
import { Link } from 'react-router-dom';

function BlogList({ posts }) {
  return (
    <div>
      <h2>Danh sách bài blog</h2>
      <ul>
        {posts.map(post => (
          <li key={post.id}>
            <Link to={`/blog/${post.id}`}>{post.title}</Link>
          </li>
        ))}
```

```
      </ul>
    </div>
  );
}
export default BlogList;
```

2. Tạo `src/components/BlogPost.js`:

```
import { useParams } from 'react-router-dom';

function BlogPost({ posts }) {
  const { postId } = useParams();
  const post = posts.find(p => p.id === parseInt(postId));

  if (!post) return <p>Bài viết không tồn tại</p>;

  return (
    <div>
      <h2>{post.title}</h2>
      <p>{post.content}</p>
    </div>
  );
}
export default BlogPost;
```

3. Cập nhật `src/App.jsx`:

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import BlogList from './components/BlogList';
import BlogPost from './components/BlogPost';

const posts = [
  { id: 1, title: 'Bài viết 1', content: 'Nội dung bài viết 1...' },
  { id: 2, title: 'Bài viết 2', content: 'Nội dung bài viết 2...' },
  { id: 3, title: 'Bài viết 3', content: 'Nội dung bài viết 3...' },
];
```

```
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/blog" element={<BlogList posts={posts} />} />
        <Route path="/blog/:postId" element={<BlogPost posts={posts} />} />
      </Routes>
    </BrowserRouter>
  );
}
export default App;
```

Bài 3: Tạo trang Not Found (404)

- **Yêu cầu**: Tạo trang 404 cho các URL không khớp, sử dụng `<Route path="*">`.
- **Hướng dẫn**:

1. Tạo `src/components/NotFoundPage.js`:

```
function NotFoundPage() {
  return <h2>404 - Trang không tồn tại</h2>;
}
export default NotFoundPage;
```

2. Cập nhật `src/App.jsx`:

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import BlogList from './components/BlogList';
import BlogPost from './components/BlogPost';
import NotFoundPage from './components/NotFoundPage';

const posts = [
  { id: 1, title: 'Bài viết 1', content: 'Nội dung bài viết 1...' },
  { id: 2, title: 'Bài viết 2', content: 'Nội dung bài viết 2...' },
];
```

```
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/blog" element={<BlogList posts={posts} />} />
        <Route path="/blog/:postId" element={<BlogPost posts={posts} />} />
        <Route path="*" element={<NotFoundPage />} />
      </Routes>
    </BrowserRouter>
  );
}
export default App;
```

Bài 4: Tạo trang Login giả với chuyển hướng

- **Yêu cầu**: Nếu chưa đăng nhập (state `isLoggedIn`), chuyển hướng từ `/profile` về `/login` bằng `useNavigate`.
- **Hướng dẫn**:

1. Tạo `src/components/LoginPage.js`:

```
import { useNavigate } from 'react-router-dom';

function LoginPage({ setIsLoggedIn }) {
  const navigate = useNavigate();

  function handleLogin() {
    setIsLoggedIn(true);
    navigate('/profile');
  }

  return <button onClick={handleLogin}>Đăng nhập</button>;
}
export default LoginPage;
```

2. Tạo `src/components/ProfilePage.js` :

```
import { useEffect } from 'react';
import { useNavigate } from 'react-router-dom';

function ProfilePage({ isLoggedIn }) {
  const navigate = useNavigate();

  useEffect(() => {
    if (!isLoggedIn) {
      navigate('/login');
    }
  }, [isLoggedIn]);

  return <h2>Hồ sơ người dùng</h2>;
}
export default ProfilePage;
```

3. Cập nhật `src/App.jsx` :

```
import { useState } from 'react';
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import LoginPage from './components/LoginPage';
import ProfilePage from './components/ProfilePage';

function App() {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  return (
    <BrowserRouter>
      <Routes>
        <Route path="/login" element={<LoginPage setIsLoggedIn={setIsLoggedI
        <Route path="/profile" element={<ProfilePage isLoggedIn={isLoggedIn}
      </Routes>
    </BrowserRouter>
  );
```

```
}
export default App;
```

Bài 5: Mở rộng Blog với nút Quay lại

- **Yêu cầu**: Trong trang `/blog/:postId`, thêm nút "Quay lại danh sách" sử dụng `useNavigate`.
- **Hướng dẫn**:

1. Cập nhật `src/components/BlogPost.js`:

```
import { useParams, useNavigate } from 'react-router-dom';

function BlogPost({ posts }) {
  const { postId } = useParams();
  const navigate = useNavigate();
  const post = posts.find(p => p.id === parseInt(postId));

  if (!post) return <p>Bài viết không tồn tại</p>;

  return (
    <div>
      <h2>{post.title}</h2>
      <p>{post.content}</p>
      <button onClick={() => navigate('/blog')}>Quay lại danh sách</button>
    </div>
  );
}
export default BlogPost;
```

2. Đảm bảo `App.jsx` từ Bài 2 hoặc 3 đã được cấu hình đúng.