

LAB 6. REACT: STATE MANAGEMENT

Thời lượng: 4 tiết

A. Mục tiêu

Bài thực hành này được thực hiện sau khi sinh viên hoàn tất các yêu cầu trong bài Lab

5. Sau khi hoàn thành bài thực hành này, sinh viên cần nắm:

- Tạo bố cục mới và các trang dành cho người quản trị, định tuyến URL cho các trang quản trị.
- Truyền và nhận tham số từ URL sử dụng useParams.
- Quản lý trạng thái giữa các thành phần trong ứng dụng bằng Redux.
- Xác thực biểu mẫu ở phía người dùng.
- Tùy biến giao diện trang quản trị.

Yêu cầu: Sinh viên cần hoàn thiện API ở Lab 4 và Lab 4 bổ sung trước khi thực hiện bài này. Có thể có một số API khác với thực tế trên máy của sinh viên, do đó sinh viên có thể tự điều chỉnh lại hoặc làm theo cách riêng cho phù hợp. Sinh viên tự làm phần “B. Hướng dẫn thực hành” ở nhà và nộp lên hệ thống LMS. Tại phòng Lab, sinh viên làm phần “C. Bài tập thực hành” dựa trên dự án đã hoàn thành ở phần B.

B. Hướng dẫn thực hành

1. Kiến thức cần nắm

Sau khi thực hiện bài Lab 5, sinh viên cần nắm được một số kiến thức quan trọng như:

1.1. Định nghĩa hàm

- **Khai báo hàm (function declaration):** Cách truyền thống để định nghĩa một hàm và khá tương đồng với các ngôn ngữ lập trình khác. Bắt đầu bằng từ khoá `function` sau đó là tên của hàm:

```
// Function declaration
function add(a, b) {
  console.log(a + b);
}
// Calling a function
add(2, 3); // 5
```

Lưu ý: Chúng ta có thể sử dụng hàm trước hoặc sau định nghĩa hàm (hoisting).

- **Biểu thức hàm (function expression):** Một cách khác để định nghĩa hàm nhưng cách viết giống như là khai báo hằng (gán biểu thức vào hằng):

```
// Function Expression
const add = function(a, b) {
  console.log(a+b);
}
// Calling function
add(2, 3); // 5
```

Lưu ý: Chúng ta chỉ có thể sử dụng sau khi định nghĩa hàm.

- **Hàm mũi tên (arrow function):** Xuất hiện trong phiên bản ES6 (ES2015) của JavaScript cho phép chúng ta rút gọn phần thân hàm. Chúng ta không dùng từ khóa `function` mà sử dụng mũi tên (`=>`):

```
// Single line of code
let add = (a, b) => a + b;
// Calling function
console.log(add(3, 2)) // 5
```

Ở hàm mũi tên, giá trị trả về ngụ ý (implicitly) sau mũi tên. Nếu như cần nhiều dòng hơn trong hàm mũi tên, ta có thể viết lại như sau:

```
// Multiple lines of code
let add = (a, b) => {
  return a + b;
}
// Calling function
console.log(add(3, 2)) // 5
```

Và đừng quên có từ khóa `return` trong cặp dấu ngoặc nhọn `{}` và `}` nhé.

Lưu ý: Mặc dù có vài sự khác nhau giữa các cách định nghĩa hàm như trên, nhưng trong ứng dụng React, người ta thường ưu tiên hơn ở hàm mũi tên¹. Việc sử dụng cách nào cũng chỉ là phong cách của người lập trình mà thôi.

1.2. useState và useRef

	useState	useRef
<i>Đặc điểm</i>	Dùng để theo dõi trạng thái của một thành phần hàm, gây ra kết xuất lại khi giá trị thay đổi.	Duy trì giá trị giữa các lần kết xuất, không gây ra kết xuất lại nếu giá trị thay đổi.
<i>Khởi tạo</i>	Chấp nhận một giá trị ban đầu và trả về hai giá trị: <ul style="list-style-type: none"> • Trạng thái • Hàm cập nhật 	Chấp nhận một giá trị ban đầu và trả về đối tượng thông qua thuộc tính <code>current</code> .
<i>Cập nhật đối tượng</i>	Chúng ta có thể cập nhật một thuộc tính của đối tượng mà không cần phải ghi đè toàn bộ bằng toán tử ba chấm (spread operator).	Thêm thuộc tính <code>ref</code> vào phần tử để truy cập trực tiếp trong DOM.
<i>Phạm vi sử dụng</i>	Khi muốn phần tử kết xuất lại mỗi khi trạng thái thay đổi chẳng hạn như tìm kiếm,...	Khi cần lấy giá trị đầu vào từ người dùng, truy cập phần tử DOM và giữ giá trị khi kết xuất.

2. Tạo bố cục cho trang quản trị

Trong tập tin `src/App.js`, cập nhật lại bố cục của trang người đọc như sau:

```
...
return (
  <Router>
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route path="/" element={<Index />} />
        <Route path='blog' element={<Index />} />
        <Route path='blog/Contact' element={<Contact />} />
        <Route path='blog/About' element={<About />} />
      </Route>
    </Routes>
  </Router>
)
```

¹ Nguồn: <https://stackoverflow.com/questions/49306148/why-is-arrow-syntax-preferred-over-function-declaration-for-functional-react-com>

```

        <Route path='blog/RSS' element={<RSS />} />
      </Route>
    </Routes>
    <Footer />
  </Router>
);

```

Tiếp tục cập nhật ở tập tin src/Pages/Layout.js:

```

import { Outlet } from 'react-router-dom';
import Navbar from '../Components/Navbar';
import Sidebar from '../Components/Sidebar';

const Layout = () => {
  return (
    <>
      <Navbar />
      <div className='container-fluid py-3'>
        <div className='row'>
          <div className='col-9'>
            <Outlet />
          </div>
          <div className='col-3 border-start'>
            <Sidebar />
          </div>
        </div>
      </div>
    </>
  );
};

```

```
export default Layout;
```

Với các bước thực hiện như trên, chúng ta có thể tạo riêng bố cục dành cho các trang quản trị mà không bị ảnh hưởng bởi bố cục của trang người đọc.

Trong thư mục src/Components tạo thư mục con có tên là Admin và tập tin Navbar.js lưu trong thư mục Admin với nội dung như sau:

```

import React from 'react';
import { Navbar as Nb, Nav } from 'react-bootstrap';
import {
  Link
} from 'react-router-dom';

const Navbar = () => {
  return (
    <Nb collapseOnSelect expand='sm' bg='white' variant='light'
      className='border-bottom shadow'>

```

```

    <div className='container-fluid'>
      <Nb.Brand href='/admin'>Tips & Tricks</Nb.Brand>
      <Nb.Toggle aria-controls='responsive-navbar-nav' />
      <Nb.Collapse id='responsive-navbar-nav' className='d-sm-inline-flex
justify-content-between'>
        <Nav className='mr-auto flex-grow-1'>
          <Nav.Item>
            <Link to='/admin/categories' className='nav-link text-dark'>
              Chủ đề
            </Link>
          </Nav.Item>
          <Nav.Item>
            <Link to='/admin/authors' className='nav-link text-dark'>
              Tác giả
            </Link>
          </Nav.Item>
          <Nav.Item>
            <Link to='/admin/tags' className='nav-link text-dark'>
              Thẻ
            </Link>
          </Nav.Item>
          <Nav.Item>
            <Link to='/admin/posts' className='nav-link text-dark'>
              Bài viết
            </Link>
          </Nav.Item>
          <Nav.Item>
            <Link to='/admin/comments' className='nav-link text-dark'>
              Bình luận
            </Link>
          </Nav.Item>
        </Nav>
      </Nb.Collapse>
    </div>
  </Nb>
)
}

```

```
export default Navbar;
```

Trong thư mục src/Pages tạo một thư mục con có tên là Admin và tập tin bố cục cho trang quản trị Layout.js trong Admin như sau:

```

import { Outlet } from 'react-router-dom';
import Navbar from '../../Components/Admin/Navbar';
import Footer from '../../Components/Footer';

const AdminLayout = () => {
  return (

```

```

    <>
    <Navbar />
    <div className='container-fluid py-3'>
      <Outlet />
    </div>
  </>
);
};

```

```
export default AdminLayout;
```

Chỉnh sửa tập tin src/App.js để tạo thêm URL cho trang quản trị:

```

...
import AdminLayout from './Pages/Admin/Layout';
...

<Route path='/' element={<Layout />}>
  <Route path='/' element={<Index />} />
  <Route path='blog' element={<Index />} />
  <Route path='blog/Contact' element={<Contact />} />
  <Route path='blog/About' element={<About />} />
  <Route path='blog/RSS' element={<RSS />} />
</Route>
<Route path='/admin' element={<AdminLayout />} />

</Route>

...
);
}

```

```
export default App;
```

Bây giờ sẽ tạo ra các trang cho Admin (nằm trong thư mục src/Pages/Admin):

- Authors.js
- Categories.js
- Comments.js
- Index.js
- Post/Posts.js (Thư mục con Post rồi đến Posts.js)
- Tags.js

Nội dung của tập tin Index.js như sau:

```

const Index = () => {
  return (
    <>
      <h1>Đây là khu vực dành cho người quản trị</h1>
    </>
  );
}

```

```
);  
  
}  
  
export default Index;  
Cập nhật tập tin src/App.js để hiển thị trang Index:  
  
...  
  
import AdminLayout from './Pages/Admin/Layout';  
import * as AdminIndex from './Pages/Admin/Index';  
  
...  
  
    <Route path='/admin' element={<AdminLayout />} >  
      <Route path='/admin' element={<AdminIndex.default />} />  
    </Route>  
  
...  
  );  
}
```

`export default App;`

Do ở tập tin App.js xuất hiện hai thành phần có cùng tên là Index (của trang người đọc và trang quản trị), do đó ta có thể đổi bí danh (alias) cho trang Index theo ý muốn, chẳng hạn AdminIndex. Để gọi thành phần đó trong thuộc tính `element` của thẻ Route ta thêm chữ `default` ở phía sau.

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Đây là khu vực dành cho người quản trị

© 2023 - Tips & Tricks Blog

Hình 1

Lưu ý: Sinh viên tự khai báo thành phần hàm trong các tập tin còn lại của thư mục `src/Pages/Admin`.

Gọi các thành phần và định tuyến ở tập tin App.js:

```
...  
<Route path='/admin' element={<AdminLayout />} >  
  <Route path='/admin' element={<AdminIndex.default />} />  
  <Route path='/admin/authors' element={<Authors />} />  
  <Route path='/admin/categories' element={<Categories />} />  
  <Route path='/admin/comments' element={<Comments />} />  
  <Route path='/admin/posts' element={<Posts />} />  
  <Route path='/admin/tags' element={<Tags />} />  
</Route>  
...
```

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Đây là trang bình luận

© 2023 - Tips & Tricks Blog

Hình 2

3. Tạo trang 404 Not Found, 400 Bad Request và biểu tượng Loading

3.1. Tạo trang 404

Thỉnh thoảng người dùng ghé thăm bằng một URL mà không tồn tại trên website. Thay vì người dùng nhấn vào nút Back và rời khỏi trang, chúng ta có thể sử dụng trang 404 để giúp họ điều hướng về website của chúng ta. Trong React, chúng ta sử dụng React Router để tạo một trang 404 Not Found.

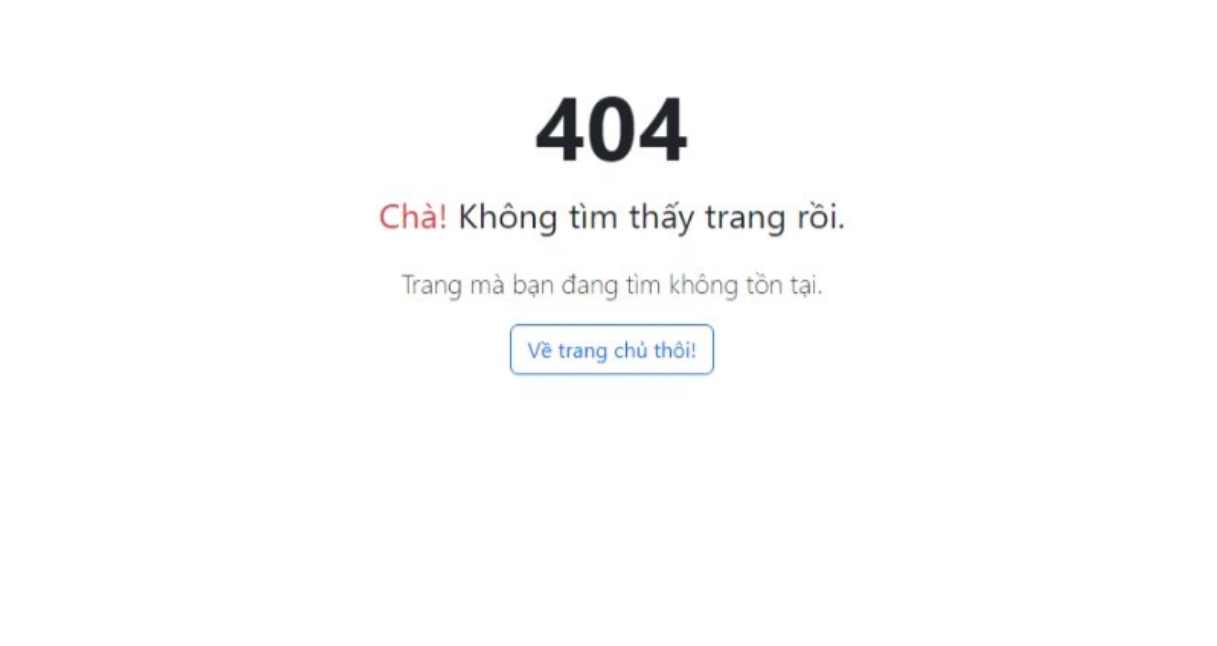
Tạo tập tin src/Pages/NotFound.js như sau:

```
...  
const NotFound = () => {  
  return (  
    <>  
      ...  
    </>  
  );  
}
```



```
export default NotFound;
```

Lưu ý: Sinh viên tự thiết kế giao diện trang 404 Not Found và nhập các thành phần cần thiết ở phía trên:



© 2023 - Tips & Tricks Blog

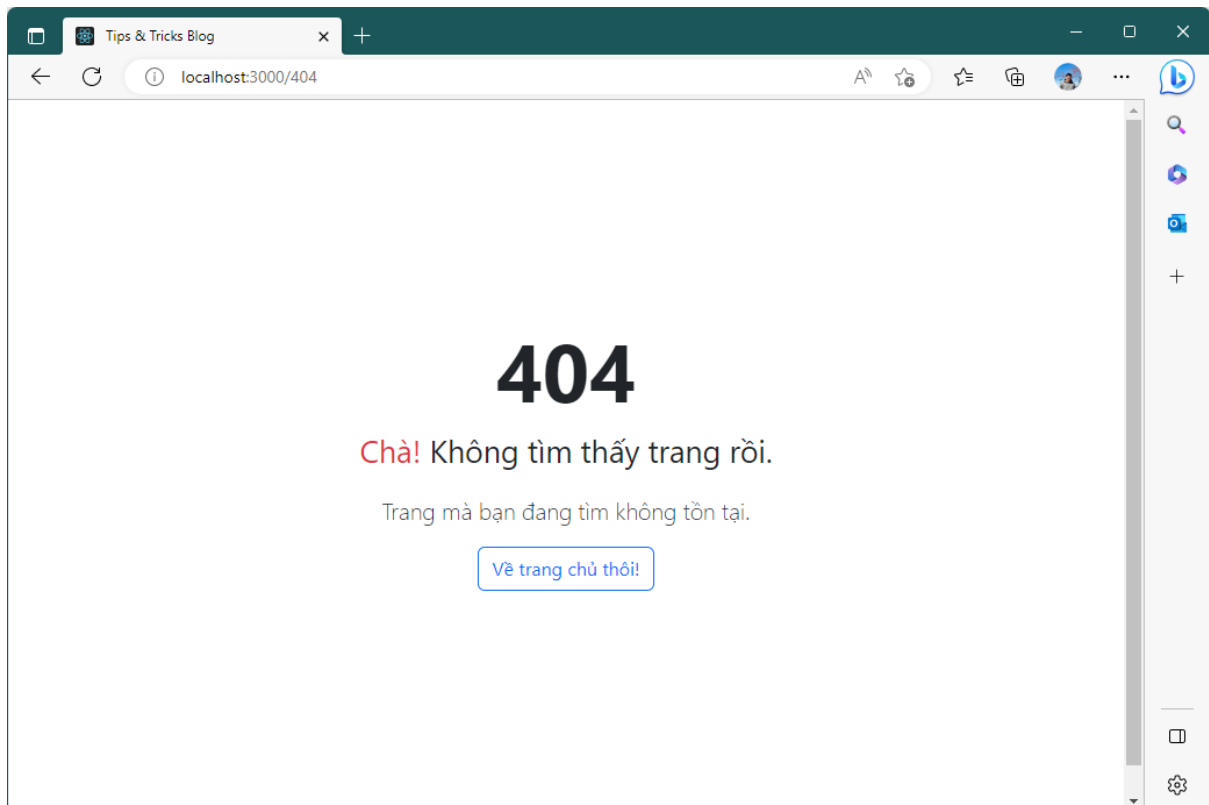
Hình 3

Để định tuyến đến trang 404 cho đường dẫn không tồn tại, bổ sung ở tập tin `arc/App.js`:

```
...
import NotFound from './Pages/NotFound';
...
return (
  <Router>
    <Routes>
...
      <Route path='*' element={<NotFound />} />
    </Routes>
...

```

Trang 404 sẽ hiển thị ngay khi người dùng truy cập website bằng các đường dẫn không tồn tại, thay vì chỉ ra đường dẫn không tồn tại đó thì sử dụng dấu hoa thị (*).



Hình 4

3.2. Tạo trang 400

Trang 400 Bad Request hiển thị khi người dùng truy cập vào một trang với tham số trong URL không hợp lệ.

Tạo tập tin `src/Pages/BadRequest.js` nội dung như sau:

```
import { Link } from 'react-router-dom';
import { useQuery } from '../Utils/Utils';

const BadRequest = () => {

  let query = useQuery(),
      redirectTo = query.get('redirectTo') ?? '/';

  return (
    <>
      ...
    </>
  );
}

export default BadRequest;
```

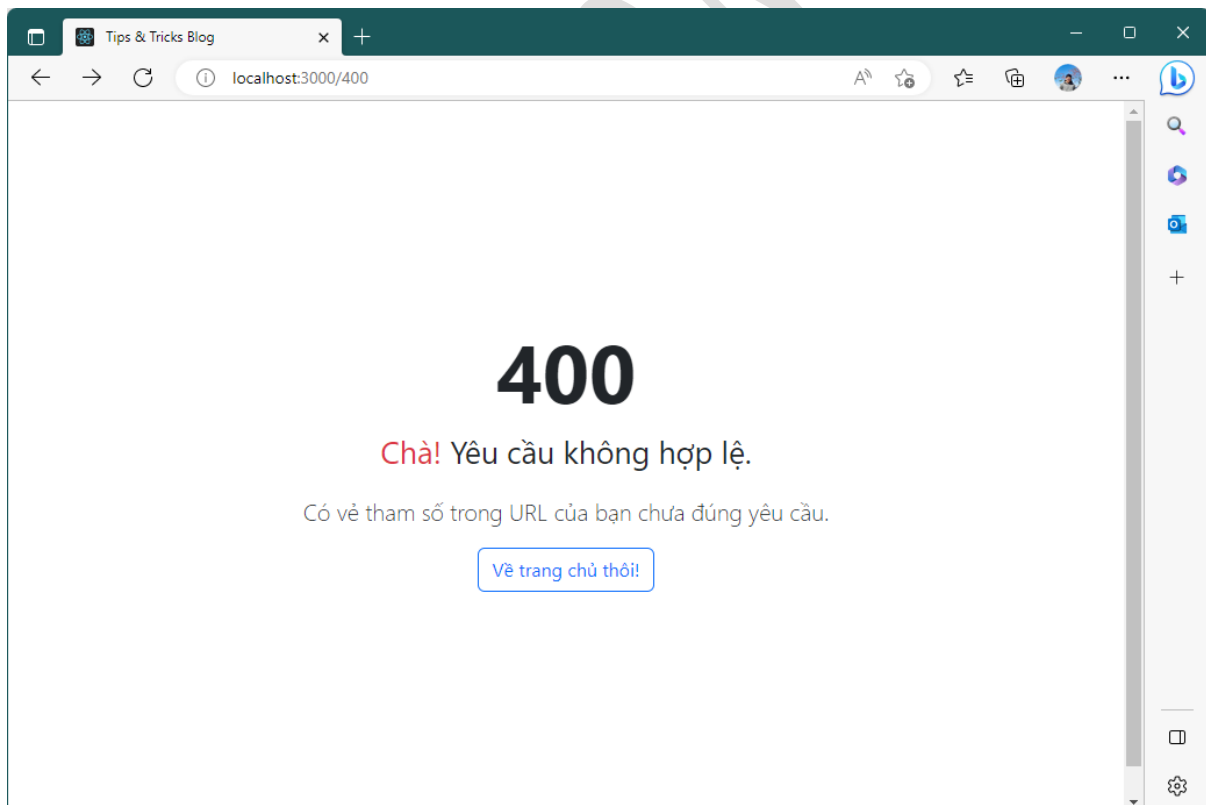
Lưu ý 1: Hàm useQuery được di chuyển ra từ src/Pages/Index.js sang src/Utils/Utils.js. Sau khi tách riêng thì nhớ nhập hàm đúng đường dẫn.

Lưu ý 2: Có thể sử dụng redirectTo để điều hướng đến trang mong muốn khi người dùng nhấn vào liên kết có trong trang này.

Tạo định tuyến cho trang 400 này ở tập tin src/App.js:

```
...
import NotFound from './Pages/NotFound';
import BadRequest from './Pages/BadRequest';
...
return (
  <Router>
    <Routes>
...
      <Route path='/400' element={<BadRequest />} />
      <Route path='*' element={<NotFound />} />
    </Routes>
  </Router>
);
...
```

Kết quả:



Hình 5

3.3. Hiển thị Loading khi lấy dữ liệu từ API

Trong quá trình chờ máy chủ API trả về dữ liệu, web cần hiển thị một thông tin nào đó để báo người dùng biết là đang tải và sẽ ẩn đi sau khi nhận được dữ liệu.

Tạo tập tin src/Components/Loading.js:

```
import Spinner from 'react-bootstrap/Spinner';
import Button from 'react-bootstrap/Button';

const Loading = () => {
  return (
    <div className='text-center'>
      <Button variant='outline-success' disabled style={{ border: 'none' }}>
        <Spinner
          as='span'
          animation='grow'
          size='sm'
          role='status'
          aria-hidden='true'
        />
        &nbsp;Đang tải...
      </Button>
    </div>
  );
}
```

```
export default Loading;
```

Trong tập tin src/Pages/Admin/Post/Posts.js, thay đổi nội dung để lấy dữ liệu từ API và hiển thị ở dạng bảng:

```
import React, { useEffect, useState } from 'react';
import Table from 'react-bootstrap/Table';
import { Link } from 'react-router-dom';
import { getPosts } from '../../../Services/BlogRepository';
import Loading from '../../../Components/Loading';

const Posts = () => {
  const [postsList, setPostsList] = useState([]);
  const [isVisibleLoading, setIsVisibleLoading] = useState(true);

  let k = '', p = 1, ps = 10;

  useEffect(() => {
    document.title = 'Danh sách bài viết';

    getPosts(k, ps, p).then(data => {
      if (data)
        setPostsList(data.items);
      else
    })
  })
}
```

```

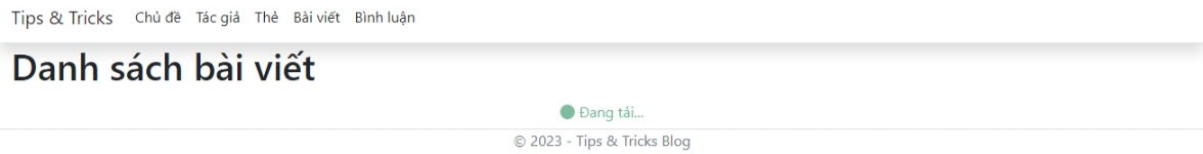
        setPostsList([]);
        setIsVisibleLoading(false);
    })
}, [k, p, ps]);

return (
    <>
    <h1>Danh sách bài viết {id}</h1>
    {isVisibleLoading ? <Loading /> :
    <Table striped responsive bordered>
        <thead>
            <tr>
                <th>Tiêu đề</th>
                <th>Tác giả</th>
                <th>Chủ đề</th>
                <th>Xuất bản</th>
            </tr>
        </thead>
        <tbody>
            {postsList.length > 0 ? postsList.map((item, index) =>
                <tr key={index}>
                    <td>
                        <Link
                            to={` /admin/posts/edit/${item.id}`}
                            className='text-bold'>
                            {item.title}
                        </Link>
                        <p className='text-muted'>{item.shortDescription}</p>
                    </td>
                    <td>{item.author.fullName}</td>
                    <td>{item.category.name}</td>
                    <td>{item.published ? "Có" : "Không"}</td>
                </tr>
            ) :
            <tr>
                <td colspan={4}>
                    <h4 className='text-danger text-center'>Không tìm thấy bài
viết nào</h4>
                </td>
            </tr>
        </tbody>
    </Table>
    }
    </>
    );
}

export default Posts;

```

Kết quả:



Hình 6

Tips & Tricks	Chủ đề	Tác giả	Thẻ	Bài viết	Bình luận
Danh sách bài viết					
Tiêu đề		Tác giả	Chủ đề	Xuất bản	
JWT creation and validation in Python using Authlib Authlib is a Python library that provides various OAuth, OpenID Connect, and JWT functionality. Authlib is my preferred library for JWT functionality, as it is one of the better Python implementations for JWT best practices, designed with OAuth and OpenID Connect in mind.		Jason Mouth	Domain Driven Design	Không	
C# Code Rules The C# Compiler's name is Roslyn. Roslyn has a very large set of analyzers to check the quality of your code, but you must turn these analyzers on before they start doing anything. This post gives you some quick information on why it's important to turn these analyzers on in your C# projects, how to do that, and how to configure them.		Kathy Smith	Practices	Không	
Performance Vs. Scalability Performance and scalability are two related but separate aspects of a system. However, there is a lot of confusion around the two terms, which often leads to architectural mistakes. This article talks about the difference between the two concepts and how to improve them.		Jessica Wonder	OOP	Không	
Papilio: An Intro Flutter gives you a powerful toolset for building rich cross-platform apps. You can build single-source apps on macOS, Windows or Linux and run those apps on phones, desktops, and in the browser. Dart is an elegant, modern language that lets you build fast, responsive, and maintainable apps. It's also familiar to Java and C# developers.		Jason Mouth	Azure	Không	
Google Cloud Run Google Cloud Run is a serverless container app service. You can deploy containerised apps to the cloud, which will autoscale horizontally with minimal configuration. It is an alternative to Kubernetes, but you only pay for usage. You do not pay for server uptime when there is no server usage.		Jason Mouth	OOP	Không	

Hình 7

Lưu ý: Sinh viên cần thận khi kết hợp điều kiện và thẻ HTML trong dòng trả về `JSX.Element`.

4. Tạo PostFilterPane (tìm kiếm)

4.1. Tối ưu gọi API

Để làm cho các dòng mã trở nên gọn gàng hơn, ta tổ chức lại các tập tin ở thư mục src/Services như sau:

- Trong Methods.js: Chứa các hàm gọi tới các phương thức của API như: GET, POST, PUT, DELETE.

```
import axios from 'axios';

export async function get_api(your_api) {
  try {
    const response = await axios.get(your_api);
    const data = response.data;
    if (data.isSuccess)
      return data.result;
    else
      return null;
  } catch (error) {
    console.log('Error', error.message);
    return null;
  }
}
```

// Các phương thức post, put, delete sẽ bổ sung sau

- Trong BlogRepository.js:

```
import { get_api } from './Methods';

export function getPosts(keyword = '',
  pageSize = 10,
  pageNumber = 1,
  sortColumn = '',
  sortOrder = '') {
  return
  get_api(`https://localhost:7085/api/posts?keyword=${keyword}&PageSize=${page
  Size}&PageNumber=${pageNumber}&SortColumn=${sortColumn}&SortOrder=${sortOrde
  r}`);
}

export function getAuthors(name = '',
  pageSize = 10,
  pageNumber = 1,
  sortColumn = '',
  sortOrder = '') {
  return
  get_api(`https://localhost:7085/api/authors?name=${name}&PageSize=${pageSize
  }&PageNumber=${pageNumber}&SortColumn=${sortColumn}&SortOrder=${sortOrder}`)
  ;
}
```

```
export function getFilter() {  
  return get_api('https://localhost:7085/api/posts/get-filter');  
}
```

- Trong Widgets.js:

```
import { get_api } from './Methods';  
  
export function getCategories() {  
  return get_api('https://localhost:7085/api/categories');  
}
```

Lưu ý: Sử dụng cú pháp JSDoc để tạo tài liệu cho tất cả các hàm ở trên: tên hàm/cách sử dụng, tham số, giá trị trả về. Xem thêm tại <https://en.wikipedia.org/wiki/JSDoc>

4.2. Tạo PostFilterPane

PostFilterPane là thanh tìm kiếm xuất hiện trên danh sách bài viết. Người quản trị có thể tìm kiếm hoặc lọc bài viết.

Tạo tập tin src/Components/Admin/PostFilterPane.js với nội dung như sau:

```
import { useState, useEffect } from 'react';  
import Form from 'react-bootstrap/Form';  
import Button from 'react-bootstrap/Button';  
import { Link } from 'react-router-dom';  
import { getFilter } from '../../Services/BlogRepository';  
  
const PostFilterPane = () => {  
  const current = new Date(),  
    [keyword, setKeyword] = useState(''),  
    [authorId, setAuthorId] = useState(''),  
    [categoryId, setCategoryId] = useState(''),  
    [year, setYear] = useState(current.getFullYear()),  
    [month, setMonth] = useState(current.getMonth()),  
    [postFilter, setPostFilter] = useState({  
      authorList: [],  
      categoryList: [],  
      monthList: []  
    });  
  
  const handleSubmit = (e) => {  
    e.preventDefault();  
  };  
  
  useEffect(() => {  
    getFilter().then(data => {  
      if (data) {  
        setPostFilter({  
          authorList: data.authorList,  

```



```

        categoryList: data.categoryList,
        monthList: data.monthList
    });
} else {
    setPostFilter({
        authorList: [],
        categoryList: [],
        monthList: []
    });
}
})
}, [])

return (
    <Form method='get'
        onSubmit={handleSubmit}
        className='row gy-2 gx-3 align-items-center p-2'>
        <Form.Group className='col-auto'>
            <Form.Label className='visually-hidden'>
                Keyword
            </Form.Label>
            <Form.Control
                type='text'
                placeholder='Nhập từ khóa...'
                name='keyword'
                value={keyword}
                onChange={e => setKeyword(e.target.value)} />
        </Form.Group>
        <Form.Group className='col-auto'>
            <Form.Label className='visually-hidden'>
                AuthorId
            </Form.Label>
            <Form.Select name='authorId'
                value={authorId}
                onChange={e => setAuthorId(e.target.value)}
                title='Author Id'
            >
                <option value=''>-- Chọn tác giả --</option>
                {postFilter.authorList.length > 0 &&
                    postFilter.authorList.map((item, index) =>
                        <option key={index} value={item.value}>{item.text}</option>
                    )}
            </Form.Select>
        </Form.Group>
        <Form.Group className='col-auto'>
            <Form.Label className='visually-hidden'>
                CategoryId
            </Form.Label>

```

```

<Form.Select name='categoryId'
  value={categoryId}
  onChange={e => setCategoryId(e.target.value)}
  title='Category Id'
>
  <option value=''>-- Chọn chủ đề --</option>
  {postFilter.categoryList.length > 0 &&
    postFilter.categoryList.map((item, index) =>
      <option key={index} value={item.value}>{item.text}</option>
    )}
</Form.Select>
</Form.Group>
<Form.Group className='col-auto'>
  <Form.Label className='visually-hidden'>
    Year
  </Form.Label>
  <Form.Control
    type='number'
    placeholder='Nhập năm...'
    name='year'
    value={year}
    max={year}
    onChange={e => setYear(e.target.value)}
  />
</Form.Group>
<Form.Group className='col-auto'>
  <Form.Label className='visually-hidden'>
    Month
  </Form.Label>
  <Form.Select name='month'
    value={month}
    onChange={e => setMonth(e.target.value)}
    title='Month'
  >
    <option value=''>-- Chọn tháng --</option>
    {postFilter.monthList.length > 0 &&
      postFilter.monthList.map((item, index) =>
        <option key={index} value={item.value}>{item.text}</option>
      )}
  </Form.Select>
</Form.Group>
<Form.Group className='col-auto'>
  <Button variant='primary' type='submit'>
    Tìm/lọc
  </Button>
  <Link to='/admin/posts/edit' className='btn btn-success ms-2'>Thêm
mới</Link>
</Form.Group>

```

```

    </Form>
  );
}

```

```
export default PostFilterPane;
```

Quay lại tập tin src/Pages/Admin/Posts.js thêm thành phần trên vào:

```

...
import PostFilterPane from '../Components/Admin/PostFilterPane';
...

return (
  <>
    <h1>Danh sách bài viết</h1>
    <PostFilterPane />
    {isVisibleLoading ? <Loading /> :

```

Kết quả:

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Danh sách bài viết

<input type="text" value="Nhập từ khóa..."/>	<input type="text" value="-- Chọn tác giả --"/>	<input type="text" value="-- Chọn chủ đề --"/>	<input type="text" value="Nhập năm..."/>	<input type="text" value="-- Chọn tháng --"/>	<input type="button" value="Xoá lọc"/> <input type="button" value="Thêm mới"/>
Tiêu đề	Tác giả	Chủ đề	Xuất bản		
JWT creation and validation in Python using Authlib Authlib is a Python library that provides various OAuth, OpenID Connect, and JWT functionality. Authlib is my preferred library for JWT functionality, as it is one of the better Python implementations for JWT best practices, designed with OAuth and OpenID Connect in mind.	Jason Mouth	Domain Driven Design	Có		
C# Code Rules The C# Compiler's name is Roslyn. Roslyn has a very large set of analyzers to check the quality of your code, but you must turn these analyzers on before they start doing anything. This post gives you some quick information on why it's important to turn these analyzers on in your C# projects, how to do that, and how to configure them.	Kathy Smith	Practices	Có		

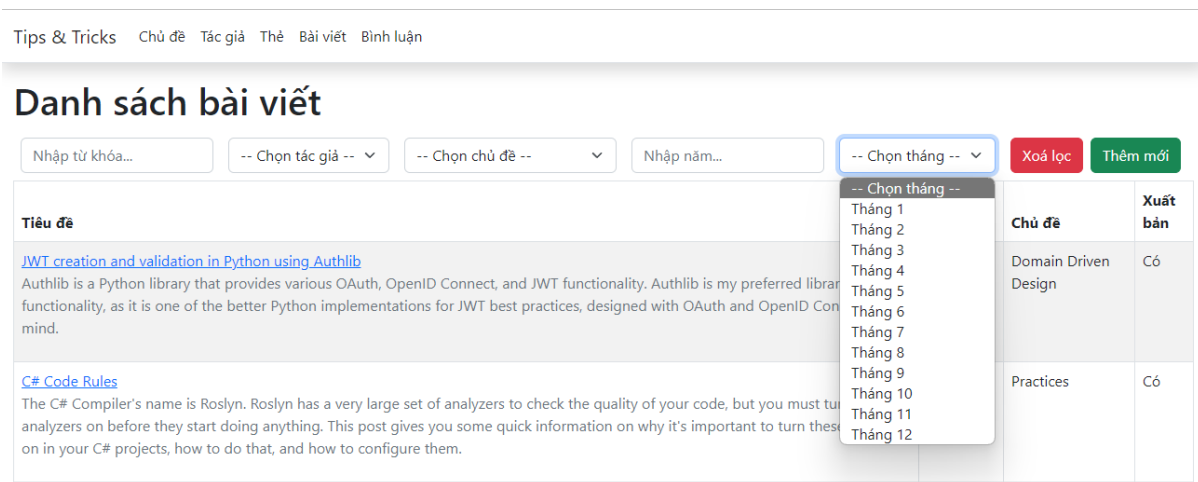
Hình 8

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Danh sách bài viết

<input type="text" value="Nhập từ khóa..."/>	<input type="text" value="-- Chọn tác giả --"/>	<input type="text" value="-- Chọn chủ đề --"/>	<input type="text" value="Nhập năm..."/>	<input type="text" value="-- Chọn tháng --"/>	<input type="button" value="Xoá lọc"/> <input type="button" value="Thêm mới"/>
Tiêu đề	Tác giả	Chủ đề	Xuất bản		
JWT creation and validation in Python using Authlib Authlib is a Python library that provides various OAuth, OpenID Connect, and JWT functionality. Authlib is my preferred library for JWT functionality, as it is one of the better Python implementations for JWT best practices, designed with OAuth and OpenID Connect in mind.	Jason Mouth	Domain Driven Design	Có		
C# Code Rules The C# Compiler's name is Roslyn. Roslyn has a very large set of analyzers to check the quality of your code, but you must turn these analyzers on before they start doing anything. This post gives you some quick information on why it's important to turn these analyzers on in your C# projects, how to do that, and how to configure them.	Kathy Smith	Practices	Có		

Hình 9



Hình 10

5. Quản lý trạng thái

Khi tới phần này có phát sinh một vấn đề như sau: *Làm sao để cập nhật được danh sách bài viết của trang Posts.js khi thành phần PostFilterPane thay đổi giá trị mà không cần thêm tham số vào URL (SearchForm tìm kiếm và hiển thị kết quả ở URL mới)?* Có một vài cách để thực hiện công việc này và đây sẽ trình bày cách sử dụng React Redux để quản lý trạng thái của PostFilterPane.

5.1. Cài đặt Redux

Cài đặt React Redux vào dự án web:

```
npm install @reduxjs/toolkit react-redux
```

Lưu ý: Dừng chương trình trước khi chạy lệnh trên.

Tiếp theo tạo một thư mục src/Redux và hai tập tin bên trong là:

- Reducer.js: Chứa các giá trị khởi tạo và reducer. `createSlice` là một hàm yêu cầu tên để nhận dạng, một trạng thái khởi tạo và một hoặc nhiều hàm reducer định nghĩa cách mà trạng thái có thể được thay đổi.

```
import { createSlice } from '@reduxjs/toolkit';
```

```
const initialState = {
  keyword: '',
  authorId: '',
  categoryId: '',
  year: '',

```

```
    month: ''
  };

const postFilterReducer = createSlice({
  name: 'postFilter',
  initialState,
  reducers: {
    reset: (state, action) => {
      return initialState;
    },
    updateKeyword: (state, action) => {
      return {
        ...state,
        keyword: action.payload
      };
    },
    updateAuthorId: (state, action) => {
      return {
        ...state,
        authorId: action.payload
      };
    },
    updateCategoryId: (state, action) => {
      return {
        ...state,
        categoryId: action.payload
      };
    },
    updateMonth: (state, action) => {
      return {
        ...state,
        month: action.payload
      };
    },
    updateYear: (state, action) => {
      return {
        ...state,
        year: action.payload
      };
    },
  },
});

export const {
  reset,
  updateKeyword,
  updateAuthorId,
  updateCategoryId,
```

```
updateMonth,  
updateYear } = postFilterReducer.actions;
```

```
export const reducer = postFilterReducer.reducer;
```

Lưu ý: Các trạng thái trong Redux là “bất biến (immutably)”, nhưng Redux Toolkit sử dụng thư viện Immer bên trong để cho phép chúng ta viết các cập nhật logic “khả biến (mutation)” bằng cách tạo ra bản sao của giá trị ban đầu:

```
return {  
  ...state,  
  keyword: action.payload.keyword  
};
```

- Store.js: Tạo một kho chứa (store) và truyền reducer ở trên vào.

```
import { configureStore } from '@reduxjs/toolkit';  
import { reducer } from './Reducer';
```

```
const store = configureStore({  
  reducer: {  
    postFilter: reducer,  
  },  
});
```

```
export default store;
```

Để React Redux <Provider> bao xung quanh ứng dụng React <App> ở tập tin src/index.js:

```
...  
import { Provider } from 'react-redux';  
import store from './Redux/Store';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
  <React.StrictMode>  
    <Provider store={store}>  
      <App />  
    </Provider>  
  </React.StrictMode>  
>);  
...
```

5.2. Sử dụng Dispatch và Selector

Cập nhật lại PostFilterPane.js:

```
import { getFilter } from '../../../Services/BlogRepository';  
import { useSelector, useDispatch } from 'react-redux';  
import {  
  reset,
```

```
    updateAuthorId,
    updateCategoryId,
    updateKeyword,
    updateMonth,
    updateYear
  } from '../Redux/Reducer';
  ...
  const PostFilterPane = () => {
    const postFilter = useSelector(state => state.postFilter),
    dispatch = useDispatch(),
    [filter, setFilter] = useState({
      authorList: [],
      categoryList: [],
      monthList: []
    });

    const handleReset = (e) => {
      dispatch(reset());
    };
    ...
    return (
      <Form method='get'
        onReset={handleReset}
        className='row gy-2 gx-3 align-items-center p-2'>
        <Form.Group className='col-auto'>
          <Form.Label className='visually-hidden'>
            Keyword
          </Form.Label>
          <Form.Control
            type='text'
            placeholder='Nhập từ khóa...'
            name='keyword'
            value={postFilter.keyword}
            onChange={e => dispatch(updateKeyword(e.target.value))} />
        </Form.Group>
        <Form.Group className='col-auto'>
          <Form.Label className='visually-hidden'>
            AuthorId
          </Form.Label>
          <Form.Select name='authorId'
            value={postFilter.authorId}
            onChange={e => dispatch(updateAuthorId(e.target.value))}
            title='Author Id'
          >
            <option value=''>-- Chọn tác giả --</option>
            {filter.authorList.length > 0 &&
              filter.authorList.map((item, index) =>
                <option key={index} value={item.value}>{item.text}</option>
              )}
          </Form.Select>
        </Form.Group>
      </Form>
    );
  };
}
```

```

    })
  </Form.Select>
</Form.Group>
<Form.Group className='col-auto'>
  <Form.Label className='visually-hidden'>
    CategoryId
  </Form.Label>
  <Form.Select name='categoryId'
    value={postFilter.categoryId}
    onChange={e => dispatch(updateCategoryId(e.target.value))}
    title='Category Id'
  >
    <option value=''>-- Chọn chủ đề --</option>
    {filter.categoryList.length > 0 &&
      filter.categoryList.map((item, index) =>
        <option key={index} value={item.value}>{item.text}</option>
      )}
  </Form.Select>
</Form.Group>
<Form.Group className='col-auto'>
  <Form.Label className='visually-hidden'>
    Year
  </Form.Label>
  <Form.Control
    type='number'
    placeholder='Nhập năm...'
    name='year'
    value={postFilter.year}
    max={postFilter.year}
    onChange={e => dispatch(updateYear(e.target.value))}
  />
</Form.Group>
<Form.Group className='col-auto'>
  <Form.Label className='visually-hidden'>
    Month
  </Form.Label>
  <Form.Select name='month'
    value={postFilter.month}
    onChange={e => dispatch(updateMonth(e.target.value))}
    title='Month'
  >
    <option value=''>-- Chọn tháng --</option>
    {filter.monthList.length > 0 && filter.monthList.map((item, index)
=>
      <option key={index} value={item.value}>{item.text}</option>
    )}
  </Form.Select>
</Form.Group>

```



```

    <Form.Group className='col-auto'>
      <Button variant='danger' type='reset'>
        Xóa lọc
      </Button>
      <Link to='/admin/posts/edit' className='btn btn-success ms-2'>Thêm
mới</Link>
    </Form.Group>
  </Form>
);
}
...

```

Bổ sung thêm hàm sau ở tập tin src/Services/BlogRepository.js:

```

export function getPostsFilter(keyword = '', authorId = '', categoryId = '',
year = '', month = '', pageSize = 10, pageNumber = 1, sortColumn = '',
sortOrder = '') {
  let url = new URL('https://localhost:7085/api/posts/get-posts-filter');
  keyword !== '' && url.searchParams.append('Keyword', keyword);
  authorId !== '' && url.searchParams.append('AuthorId', authorId);
  categoryId !== '' && url.searchParams.append('CategoryId', categoryId);
  year !== '' && url.searchParams.append('Year', year);
  month !== '' && url.searchParams.append('Month', month);
  sortColumn !== '' && url.searchParams.append('SortColumn', sortColumn);
  sortOrder !== '' && url.searchParams.append('SortOrder', sortOrder);
  url.searchParams.append('PageSize', pageSize);
  url.searchParams.append('PageNumber', pageNumber);
  return get_api(url.href);
}

```

Cập nhật lại src/Pages/Admin/Post/Posts.js:

```

import React, { useEffect, useState } from 'react';
import Table from 'react-bootstrap/Table';
import { Link, useParams, Navigate } from 'react-router-dom';
import { getPostsFilter } from '../../../Services/BlogRepository';
import Loading from '../../../Components/Loading';
import { isInteger } from '../../../Utils/Utils';
import PostFilterPane from '../../../Components/Admin/PostFilterPane';
import { useSelector } from 'react-redux';

const Posts = () => {
  const [postsList, setPostsList] = useState([]),
    [isVisibleLoading, setIsVisibleLoading] = useState(true),
    postFilter = useSelector(state => state.postFilter);

  let { id } = useParams(),
    p = 1,
    ps = 10;

  useEffect(() => {

```

```
document.title = 'Danh sách bài viết';
getPostsFilter(postFilter.keyword,
  postFilter.authorId,
  postFilter.categoryId,
  postFilter.year,
  postFilter.month,
  ps, p).then(data => {
  if (data)
    setPostsList(data.items);
  else
    setPostsList([]);
  setIsVisibleLoading(false);
});
}, [
  postFilter.keyword,
  postFilter.authorId,
  postFilter.categoryId,
  postFilter.year,
  postFilter.month,
  p, ps
]);
```

...
Kết quả: Sau khi thay đổi bộ lọc tìm kiếm thì bảng sẽ được cập nhật lại.

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Danh sách bài viết

<input type="text" value="Nhập từ khóa..."/>	<input type="text" value="-- Chọn tác giả --"/>	<input type="text" value=".NET Core"/>	<input type="text" value="Nhập năm..."/>	<input type="text" value="-- Chọn tháng --"/>	<input type="button" value="Xoá lọc"/>	<input type="button" value="Thêm mới"/>
Tiêu đề	Tác giả	Chủ đề	Xuất bản			
Dart: Formatting and Trailing Commas Trailing commas may sound like a minor aspect of the Dart language, but they have a major impact on the formatting of your code. This article explains deterministic formatting, how trailing commas affect it, why you should use them, and how to add the dart_code_metrics package to enforce them for better formatting.	Jessica Wonder	.NET Core	Có			
ASP.NET Core Diagnostic Scenarios David and friends has a great repository filled with examples of "broken patterns" in ASP.NET Core applications. It's a fantastic learning resource with both markdown and code that covers a number of common areas when writing scalable services in ASP.NET Core. Some of the guidance is general purpose but is explained through the lens of writing web services.	Jason Mouth	.NET Core	Có			

© 2023 - Tips & Tricks Blog

Hình 11

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Danh sách bài viết

<input type="text" value="Nhập từ khóa..."/>	<input type="text" value="Jason Mouth"/>	<input type="text" value=".NET Core"/>	<input type="text" value="Nhập năm..."/>	<input type="text" value="-- Chọn tháng --"/>	<input type="button" value="Xóa lọc"/>	<input type="button" value="Thêm mới"/>
Tiêu đề	Tác giả	Chủ đề	Xuất bản			
ASP.NET Core Diagnostic Scenarios David and friends has a great repository filled with examples of "broken patterns" in ASP.NET Core applications. It's a fantastic learning resource with both markdown and code that covers a number of common areas when writing scalable services in ASP.NET Core. Some of the guidance is general purpose but is explained through the lens of writing web services.	Jason Mouth	.NET Core	Có			

© 2023 - Tips & Tricks Blog

Hình 12

6. Tạo trang chỉnh sửa bài viết

6.1. Tạo trang Edit và truyền tham số id trong URL

Để kiểm tra tham số id từ đường dẫn trên và hiển thị lỗi nếu như id không phải là một số nguyên hợp lệ, bổ sung trong tập tin src/Utils/Utils.js:

```
export function isInteger(str) {  
    return Number.isInteger(Number(str)) && Number(str) >= 0;  
}
```

Bổ sung hàm mới vào tập tin src/Services/Methods.js:

```
export async function post_api(your_api, formData) {  
    try {  
        const response = await axios.post(your_api, formData);  
        const data = response.data;  
        if (data.isSuccess)  
            return data.result;  
        else  
            return null;  
    } catch (error) {  
        console.log('Error', error.message);  
        return null;  
    }  
}
```

Tiếp tục tạo một hàm mới có chức năng thêm/cập nhật bài viết ở src/Services/BlogRepository.js:

```
import { get_api, post_api } from './Methods';  
...  
export async function getPostById(id = 0) {
```

```
    if (id > 0)
      return get_api(`https://localhost:7085/api/posts/${id}`);
    return null;
  }

  export function addOrUpdatePost(formData){
    return post_api('https://localhost:7085/api/posts', formData);
  }
```

Lưu ý: Sinh viên cần kiểm tra API thêm/cập nhật bài viết đã được chỉnh sửa để chấp nhận một FormBody và FormFile giống như hướng dẫn Lab 4 bổ sung.

Thêm hàm giải mã HTML Entities ở src/Utils/Utils.js:

```
export function decode(str) {
  let txt = new DOMParser().parseFromString(str, "text/html");
  return txt.documentElement.textContent;
}
```

Tạo tập tin src/Pages/Admin/Post/Edit.js nội dung như sau:

```
import React, { useEffect, useState } from 'react';
import { isInteger, decode } from '../../../Utils/Utils';
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button';
import { Link, useParams, Navigate } from 'react-router-dom';
import { isEmptyOrSpaces } from '../../../Utils/Utils';
import { addOrUpdatePost, getFilter, getPostById } from
  '../../../Services/BlogRepository';

const Edit = () => {
  const initialState = {
    id: 0,
    title: '',
    shortDescription: '',
    description: '',
    urlSlug: '',
    meta: '',
    imageUrl: '',
    category: {},
    author: {},
    tags: [],
    selectedTags: '',
    published: false,
  },
  [post, setPost] = useState(initialState),
  [filter, setFilter] = useState({
    authorList: [],
    categoryList: [],
  });
```

```
let { id } = useParams();
id = id ?? 0;

useEffect(() => {
  document.title = 'Thêm/cập nhật bài viết';

  getPostById(id).then(data => {
    if (data)
      setPost({
        ...data,
        selectedTags: data.tags.map(tag => tag?.name).join('\r\n'),
      });
    else
      setPost(initialState);
  });

  getFilter().then(data => {
    if (data)
      setFilter({
        authorList: data.authorList,
        categoryList: data.categoryList,
      });
    else
      setFilter({
        authorList: [],
        categoryList: [],
      });
  })
}, [])

const handleSubmit = (e) => {
  e.preventDefault();
  let form = new FormData(e.target);
  addOrUpdatePost(form).then(data => {
    if (data)
      alert('Đã lưu thành công!');
    else
      alert('Đã xảy ra lỗi!');
  });
}

if (id && !isInteger(id))
  return (
    <Navigate to={`/400?redirectTo=/admin/posts`} />
  )
return (
  <>
    <Form
```

```
method='post'
encType='multipart/form-data'
onSubmit={handleSubmit}
>
<Form.Control type='hidden' name='id' value={post.id} />
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Tiêu đề
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Control
      type='text'
      name='title'
      title='Title'
      required
      value={post.title || ''}
      onChange={e => setPost({
        ...post,
        title: e.target.value
      })}
    />
  </div>
</div>
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Slug
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Control
      type='text'
      name='urlSlug'
      title='Url slug'
      value={post.urlSlug || ''}
      onChange={e => setPost({
        ...post,
        urlSlug: e.target.value
      })}
    />
  </div>
</div>
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Giới thiệu
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Control
      as='textarea'
      type='text'
```

```
        required
        name='shortDescription'
        title='Short description'
        value={decode(post.shortDescription || '')}
        onChange={e => setPost({
            ...post,
            shortDescription: e.target.value
        })}
    />
</div>
</div>
<div className='row mb-3'>
    <Form.Label className='col-sm-2 col-form-label'>
        Nội dung
    </Form.Label>
    <div className='col-sm-10'>
        <Form.Control
            as='textarea'
            rows={10}
            type='text'
            required
            name='description'
            title='Description'
            value={decode(post.description || '')}
            onChange={e => setPost({
                ...post,
                description: e.target.value
            })}
        />
    </div>
</div>
<div className='row mb-3'>
    <Form.Label className='col-sm-2 col-form-label'>
        Metadata
    </Form.Label>
    <div className='col-sm-10'>
        <Form.Control
            type='text'
            name='meta'
            title='meta'
            required
            value={decode(post.meta || '')}
            onChange={e => setPost({
                ...post,
                meta: e.target.value
            })}
        />
    </div>
</div>
```

```
</div>
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Tác giả
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Select
      name='authorId'
      title='Author Id'
      value={post.author.id}
      required
      onChange={e => setPost({
        ...post,
        author: e.target.value
      })}
    >
      <option value=''>-- Chọn tác giả --</option>
      {filter.authorList.length > 0 &&
        filter.authorList.map((item, index) =>
          <option key={index}
value={item.value}>{item.text}</option>
        )}
    </Form.Select>
  </div>
</div>
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Chủ đề
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Select
      name='categoryId'
      title='Category Id'
      required
      value={post.category.id}
      onChange={e => setPost({
        ...post,
        category: e.target.value
      })}
    >
      <option value=''>-- Chọn chủ đề --</option>
      {filter.categoryList.length > 0 &&
        filter.categoryList.map((item, index) =>
          <option key={index}
value={item.value}>{item.text}</option>
        )}
    </Form.Select>
  </div>
</div>
```



```

</div>
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Từ khóa (mỗi từ 1 dòng)
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Control as='textarea'
      rows={5}
      type='text'
      name='selectedTags'
      title='Selected Tags'
      required
      value={post.selectedTags}
      onChange={e => setPost({
        ...post,
        selectedTags: e.target.value
      })}
    >
  </Form.Control>
</div>
</div>
{!isEmptyOrSpaces(post.imageUrl) && <div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Hình hiện tại
  </Form.Label>
  <div className='col-sm-10'>
    <img src={post.imageUrl} alt={post.title} />
  </div>
</div>
}
<div className='row mb-3'>
  <Form.Label className='col-sm-2 col-form-label'>
    Chọn hình ảnh
  </Form.Label>
  <div className='col-sm-10'>
    <Form.Control
      type='file'
      name='imageFile'
      accept='image/*'
      title='Image file'
      onChange={e => setPost({
        ...post,
        imageFile: e.target.files[0]
      })}
    />
  </div>
</div>
<div className='row mb-3'>

```

```

    <div className='col-sm-10 offset-sm-2'>
      <div className='form-check'>
        <input
          className='form-check-input'
          type='checkbox'
          name='published'
          checked={post.published}
          title='Published'
          onChange={e => setPost({
            ...post,
            published: e.target.checked
          })}
        />
        <Form.Label className='form-check-label'>
          Đã xuất bản
        </Form.Label>
      </div>
    </div>
  </div>
  <div className='text-center'>
    <Button variant='primary' type='submit'>
      Lưu các thay đổi
    </Button>
    <Link to='/admin/posts' className='btn btn-danger ms-2'>
      Hủy và quay lại
    </Link>
  </div>
</Form>
</>
);
}

```

`export default Edit;`

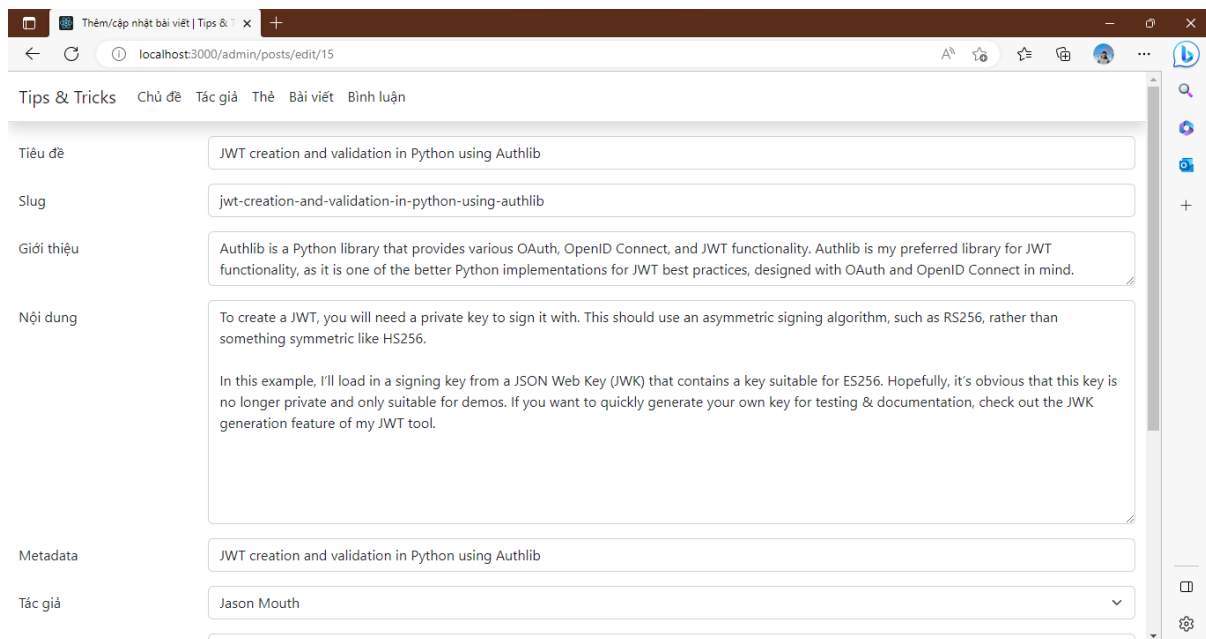
Trong tập tin `src/App.js`, thêm hai thẻ Route mới, trong đó có một thẻ Route có tham số id trong URL:

```

...
<Route path='/admin' element={<AdminIndex.default />} />
<Route path='/admin/authors' element={<Authors />} />
<Route path='/admin/categories' element={<Categories />} />
<Route path='/admin/comments' element={<Comments />} />
<Route path='/admin/posts' element={<Posts />} />
<Route path='/admin/posts/edit' element={<Edit />} />
<Route path='/admin/posts/edit/:id' element={<Edit />} />
<Route path='/admin/tags' element={<Tags />} />
...

```

Kết quả:



Thêm/cập nhật bài viết | Tips & Tricks

localhost:3000/admin/posts/edit/15

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Tiêu đề: JWT creation and validation in Python using Authlib

Slug: jwt-creation-and-validation-in-python-using-authlib

Giới thiệu: Authlib is a Python library that provides various OAuth, OpenID Connect, and JWT functionality. Authlib is my preferred library for JWT functionality, as it is one of the better Python implementations for JWT best practices, designed with OAuth and OpenID Connect in mind.

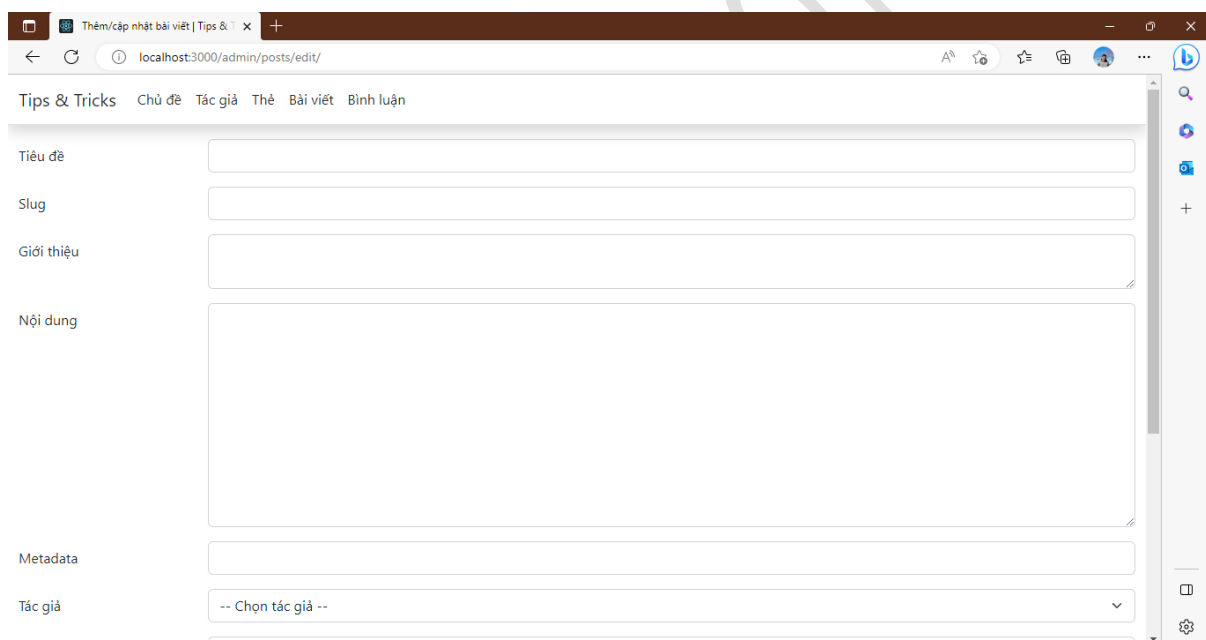
Nội dung: To create a JWT, you will need a private key to sign it with. This should use an asymmetric signing algorithm, such as RS256, rather than something symmetric like HS256.

In this example, I'll load in a signing key from a JSON Web Key (JWK) that contains a key suitable for ES256. Hopefully, it's obvious that this key is no longer private and only suitable for demos. If you want to quickly generate your own key for testing & documentation, check out the JWK generation feature of my JWT tool.

Metadata: JWT creation and validation in Python using Authlib

Tác giả: Jason Mouth

Hình 13



Thêm/cập nhật bài viết | Tips & Tricks

localhost:3000/admin/posts/edit/

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Tiêu đề:

Slug:

Giới thiệu:

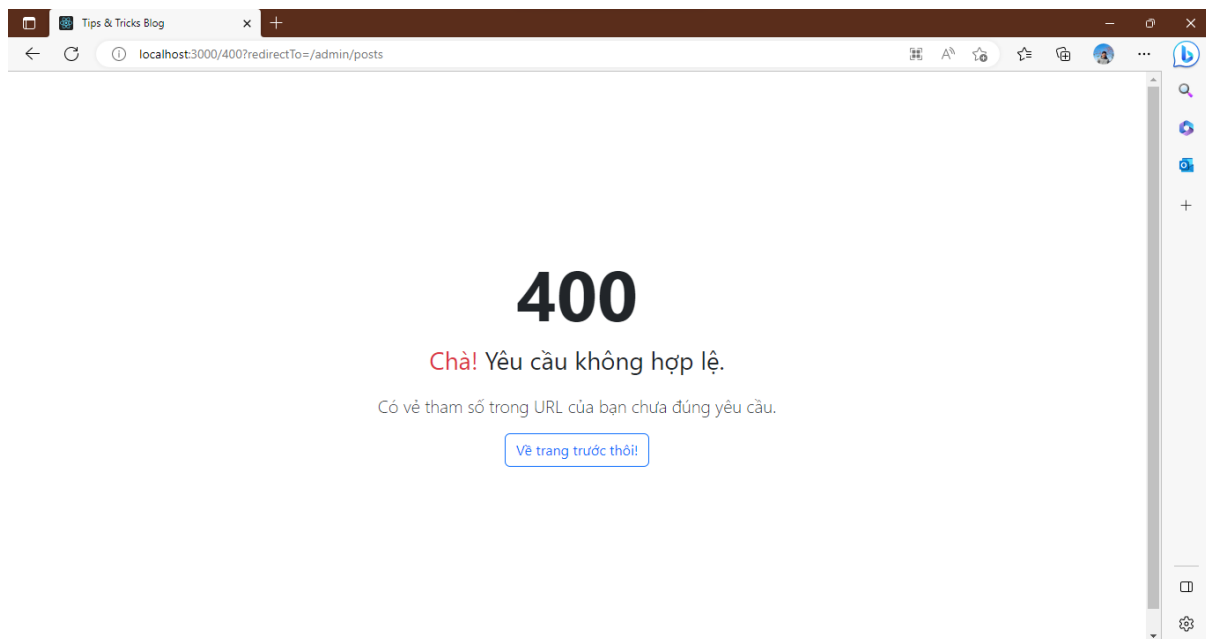
Nội dung:

Metadata:

Tác giả: -- Chọn tác giả --

Hình 14

Hoặc truy cập vào <http://localhost:3000/admin/posts/edit/a>:



Hình 15

Có thể thêm một bài viết kèm hình ảnh:

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Tiêu đề

This is a title

Slug

this-is-a-title

Giới thiệu

1

Nội dung

1

Metadata

1

Tác giả

Jason Mouth


Chủ đề

.NET Core

Từ khóa (mỗi từ 1 dòng)

1

Hình hiện tại



Chọn hình ảnh

Choose File No file chosen

☒ Đã xuất bản

Lưu các thay đổi

Hủy và quay lại

© 2023 - Tips & Tricks Blog

Hình 16

Lưu ý: Nếu như thêm thành công bài viết có hình ảnh mà web vẫn chưa hiển thị ảnh thì cần kiểm tra imageUrl có chứa địa chỉ của máy chủ API hay không (máy chủ API và máy chủ React webpack khác nhau).

6.2. Xác thực biểu mẫu

Trong tập tin `src/Pages/Admin/Post/Edit.js` sinh viên bổ sung thêm:

```
const [validated, setValidated] = useState(false);
```

Chỉnh sửa hàm `handleSubmit` như sau:

```
const handleSubmit = (e) => {
  e.preventDefault();
  if (e.currentTarget.checkValidity() === false) {
    e.stopPropagation();
    setValidated(true);
  } else {
    let form = new FormData(e.target);
    form.append('published', post.published);
    addOrUpdatePost(form).then(data => {
      if (data)
        alert('Đã lưu thành công!');
      else
        alert('Đã xảy ra lỗi!');
    });
  }
}
```

Ở thẻ `Form` bổ sung hai thuộc tính sau (nằm trong thẻ `Form` mở):

```
noValidate
validated={validated}
```

Các bước trên sử dụng xác thực dữ liệu của Bootstrap, trong đó thuộc tính `noValidate` tắt các hộp thoại chú giải (tooltip) mặc định của trình duyệt, nhưng vẫn cung cấp truy cập đến API xác thực biểu mẫu trong JavaScript.

Kế tiếp là thêm các dòng chữ cảnh báo ở tất cả các trường dữ liệu bắt buộc (để sau các thẻ `Form.Control` đóng):

```
<Form.Control.Feedback type='invalid'>
  Không được bỏ trống.
</Form.Control.Feedback>
```

Ngoài ra còn có các thuộc tính ràng buộc khác như `maxLength`, `minLength`, `max`, `min`, `disabled`, `readonly`.

Kết quả:

Tips & Tricks Chủ đề Tác giả Thẻ Bài viết Bình luận

Tiêu đề	<input type="text"/>	ⓘ
	Không được bỏ trống.	
Slug	<input type="text"/>	✓
Giới thiệu	<input type="text"/>	ⓘ
	Không được bỏ trống.	
Nội dung	<div></div>	ⓘ
	Không được bỏ trống.	
Metadata	<input type="text"/>	ⓘ
	Không được bỏ trống.	
Tác giả	-- Chọn tác giả --	ⓘ ▼
	Không được bỏ trống.	
Chủ đề	-- Chọn chủ đề --	ⓘ ▼
	Không được bỏ trống.	
Từ khóa (mỗi từ 1 dòng)	<input type="text"/>	ⓘ
	Không được bỏ trống.	
Chọn hình ảnh	<div>Choose File No file chosen</div>	✓
	<input type="checkbox"/> Đã xuất bản	

Lưu các thay đổi

Hủy và quay lại

© 2023 - Tips & Tricks Blog

Hình 17

Lưu ý: Cách xác thực dữ liệu ở phía người dùng (client-side) và các thuộc tính ràng buộc mặc định của các thẻ input không thể ngăn chặn hoàn toàn các kỹ thuật vượt qua ràng buộc của những người có kinh nghiệm. Do đó cần phải kết hợp cả ràng buộc ở phía máy chủ (server-side) để ngăn những dữ liệu không mong muốn.

C. Bài tập thực hành

1. Tiếp tục hoàn thiện các chức năng quản lý bài viết:

Trong phần hướng dẫn, trang hiển thị danh sách bài viết luôn hiển thị 10 bài viết mới nhất. Hãy cập nhật lại mã nguồn và thêm điều khiển phân trang để người quản trị có thể tải và xem tất cả các bài viết. (Sinh viên tham khảo cách xây dựng điều khiển phân trang ở bài Lab trước).

Hiện tại cột “Xuất bản” hiển thị một trong 2 giá trị: “Có”, “Không”. Hãy cập nhật lại mã nguồn để hiển thị giá trị này dưới dạng nút bấm. Khi người dùng click chuột thì đổi trạng thái Xuất bản của bài viết.

Bổ sung thêm mã lệnh để hiển thị nút xóa trên mỗi dòng ứng với bài viết. Khi người dùng click chuột vào nút này thì hỏi “Bạn có thực sự muốn xóa bài viết này không?”. Nếu người dùng trả lời Yes, thực hiện việc xóa bài viết và tải lại trang.

Trên khung tìm kiếm, bổ sung thêm điều kiện tìm kiếm “Chưa xuất bản”, hiển thị dưới dạng checkbox. Khi người dùng đánh dấu chọn checkbox này và nhấn tìm kiếm thì chỉ hiển thị những bài viết có cờ Published bằng false.

Trên khung tìm kiếm, thêm nút “Bỏ lọc”. Khi người dùng nhấn vào nút này thì xóa tất cả các điều kiện tìm kiếm trong các ô nhập và tải lại trang chứa đầy đủ bài viết.

2. Cài đặt các chức năng xem danh sách, thêm, xóa, cập nhật chủ đề.
3. Cài đặt các chức năng xem danh sách, thêm, xóa, cập nhật tác giả.
4. Cài đặt các chức năng xem danh sách, thêm, xóa, cập nhật thẻ (tag).
5. Cài đặt các chức năng xem danh sách, phê duyệt, xóa các bình luận.
6. Cài đặt các chức năng xem danh sách, quản lý người đăng ký theo dõi blog.
7. Ở trang “Bảng điều khiển”, hãy tạo các thống kê về: Tổng số bài viết, số bài viết chưa xuất bản, số lượng chủ đề, số lượng tác giả, số lượng bình luận đang chờ phê duyệt, số lượng

D. Tài liệu tham khảo

- [1] “Different ways of writing functions in JavaScript – GeeksforGeeks,” *Geeksforgeeks*, <https://www.geeksforgeeks.org/different-ways-of-writing-functions-in-javascript/>
- [2] “Quick Start | React Redux,” *React-redux*, <https://react-redux.js.org/tutorials/quick-start>
- [3] “React useState Hook,” *W3schools*, https://www.w3schools.com/react/react_usestate.asp
- [4] “React useRef Hook,” *W3schools*, https://www.w3schools.com/react/react_useref.asp

--- HẾT ---