直播第二堂

Date	2026/01/05
Select	2025 React
備註	Empty
直播助教	yuyu

RESTful API 串接

💡 記得要錄影

大綱

1. 主線任務說明
2. 實作練習
    a. 建立 API 設定
    b. 建立登入頁面
    c. 建立狀態管理
    d. 實作登入功能
    e. 建立產品管理頁面
    f. 實作產品資料載入

1. 主線任務說明

第二週開始要接觸到課程的 API，是非常重要的一個章節，這份 API 會一直跟到直播班結束，同學們可用此份 API 建立屬於自己的電商資料，請務必在本週的時間內完成。

請**使用 vite** 完成以下需求：

- 使用者可以從登入頁面登入，並轉到後台商品頁面
- 使用者若無登入直接進入商品頁面，會被導回登入頁面
- 使用者可以查看產品列表
- 使用者可以點擊單一產品，查看詳細資訊

API 申請說明

這個 API 與『Vue 直播班』是同一個資料庫，因此若已經有帳號了，也可以登入該帳號並**申請一個新的 API 路徑**就好。

若是還沒有帳號，就註冊一個來使用就可以哩～

課程 API 相關網址：

- 註冊連結、測試管理平台
- API 文件

登入頁面 API

- 登入串接 POST API
- 驗證登入串接 POST API

產品頁面 API

- 取得產品資料串接 GET API

頁面模板

- 登入到產品頁面的頁面模板（因這週尚未教到路由，所以會先用 三元運算子 切換畫面呈現）

作業須符合此作業規範

每週主線任務範例：https://github.com/hexschool/react-training-chapter-2025

挑戰等級

- LV 1｜參考程式碼範例，並重新撰寫及補上註解（禁止複製範例程式碼）
- LV 2｜僅使用課程版型，並重新撰寫產品列表的功能
- LV 3｜不使用課程版型完成此功能

登入頁面

產品頁面

專案建立

使用 Vite 建立專案

```
# 建立新專案 npm create vite@latest # 進入專案目錄 cd week2 # 安裝依賴 npm in
stall axios bootstrap@5.3.8 # 啟動開發伺服器 npm run dev
```

專案結構

```
week2/ ├── src/ │ ├── App.jsx # 主元件 │ ├── main.jsx # 應用程式入口 │ └── a
ssets/ │ └── style.css # 樣式 ├── index.html # HTML 模板 ├── package.json #
專案設定 └── vite.config.js # Vite 設定
```

流程

```
1. 使用者輸入帳密 → 呼叫登入 API 2. 登入成功 → 取得 token 3. 將 token 存到
Cookie 4. 將 token 放進 axios Authorization header 5. 使用 token 取得產品資
料 6. 根據登入狀態切換畫面
```

2. 實作步驟

建立 .env 檔案

- 變數名稱一定要以 `VITE_` 開頭
- .env 不要上傳到 GitHub

```
VITE_API_BASE=https://ec-course-api.hexschool.io/v2 VITE_API_PATH=你的API路徑
```

載入 bootstrap 的 css 與 js

```
// main.jsx import 'bootstrap/dist/css/bootstrap.min.css'; import 'bootstrap/dist/js/bootstrap.bundle.min.js';
```

建立 API 設定

```
import axios from "axios"; // API 設定 const API_BASE = import.meta.env.VITE_API_BASE; const API_PATH = import.meta.env.VITE_API_PATH;
```

引入 style

```
// App.jsx import "./assets/style.css";
```

建立登入頁面

切換完全 `isAuth` 狀態控制：

- false → 顯示登入頁
- true → 顯示產品管理頁
- forms

```
// 登入表單畫面 {!isAuth ? ( <div className="container login"> <div className
me="row justify-content-center"> <h1 className="h3 mb-3 font-weight-norma
l">請先登入</h1> <div className="col-8"> <form id="form" className="form-si
gnin" onSubmit={handleSubmit}> <div className="form-floating mb-3"> <input
type="email" className="form-control" id="username" name="username" placeh
older="name@example.com" value={formData.username} onChange={handleInputCh
ange} required autoFocus /> <label htmlFor="username">Email address</label
> </div> <div className="form-floating"> <input type="password" className
="form-control" id="password" name="password" placeholder="Password" value
={formData.password} onChange={handleInputChange} required /> <label htmlF
or="password">Password</label> </div> <button className="btn btn-lg btn-pr
imary w-100 mt-3" type="submit" > 登入 </button> </form> </div> </div> <p c
lassName="mt-5 mb-3 text-muted">&copy; 2025~∞ - 六角學院</p> </div> ) : (
// 登入後的產品管理頁面 (同第一週) )}
```

建立狀態管理

```
function App() { // 表單資料狀態(儲存登入表單輸入) const [formData, setFormDa
ta] = useState({ username: "", password: "", }); // 登入狀態管理(控制顯示登
入或產品頁)  const [isAuth, setIsAuth] = useState(false); // 產品資料狀態 co
nst [products, setProducts] = useState([]); // 目前選中的產品 const [tempPro
duct, setTempProduct] = useState(null); }
```

表單輸入處理

```
// 表單輸入處理 const handleInputChange = (e) => { const { name, value } =
e.target; setFormData((prevData) => ({ ...prevData, // 保留原有屬性 [name]:
value, // 更新特定屬性 })); };
```

Cookie 操作

設定 cookie、讀取 cookie

```
// 設定 Cookie document.cookie = `hexToken=${token};expires=${new Date(expi
red)};`; // 讀取 Cookie const token = document.cookie .split("; ") .find((r
ow) => row.startsWith("hexToken=")) ?.split("=")[1];
```

實作登入功能

在 API 中，我們通常會在 `Authorization` Header 傳送 Token

常見格式如下：

```
Authorization: Bearer xxxxxxx.yyyyyyy.zzzzzzz
```

課程使用的 API（Swagger）不需要加 Bearer，請直接傳送 `Token` 字串即可，否則會驗證失敗

設定 Authorization Header

axios 全域設置

登入成功後，請將 Token 設定到 axios 的預設 Header，之後所有 API 請求都會自動帶上 Token

```
// 修改實體建立時所指派的預設配置
axios.defaults.headers.common['Authorization'] = AUTH_TOKEN;
```

補充：axios 也可以透過 create 建立實體並設定預設 Header

```
// 建立實體時指派預設配置 const instance = axios.create({ baseURL:
'https://api.example.com' });
```

- 登入 api `${API_BASE}/admin/signin`
- 驗證 token api `${API_BASE}/api/user/check`
- 驗證

登入時設定 token，是給「本次操作」使用

checkLogin 是為了「重新整理頁面後」重新取回 token

建立產品管理頁面

整合第一週作業

產品 API 需要 token，因此必須在「登入成功後」才能呼叫

```
// 產品管理頁面 (同第一週) { isAuth && ( <div className="container"> <div cl
assName="row mt-5"> <div className="col-md-6"> {/* 產品列表區塊 (同第一週)
*/} </div> <div className="col-md-6">{/* 產品詳情區塊 (同第一週) */}</div>
</div> </div> ); }
```

實作產品資料載入

登入後取得產品資料

取得產品資料 api `${API_BASE}/api/${API_PATH}/admin/products`

```
// 取得產品資料 const getData = async () => { try { const response = await
axios.get( `${API_BASE}/api/${API_PATH}/admin/products` ); console.log("產
品資料：", response.data); setProducts(response.data.products); } catch (er
r) { console.error("取得產品失敗：", err.response?.data?.message); } };
```