RESTFUL API

林新德

shinder.lin@gmail.com

1.1 什麼是 Restful API

- REST為Representational State Transfer（表現層狀態轉換）的縮寫。
- 2000年由 Dr. Roy Thomas Fielding 在其博士論文中提出的 HTTP 資料交換風格。
- 要點：

1. 以 URI 指定資源，使用 HTTP 或 HTTPS 為操作協定。
2. 透過操作資源的表現形式來操作資料。 讀取、新增、修改、刪除
3. 就是以 HTTP 的 GET, POST, PUT, DELETE 方法對應到操作資源的 CRUD。
4. 資源的表現形式沒有限定，可以是 HTML, XML, **JSON** 或其它格式。
5. REST 是設計風格並不是標準，所以沒有硬性的規定。

- 實作 REST 的 後端 API 一般稱作 Restful API

1.2 以商品資料為說明

- 取得列表：
- http://my-domain/products (GET)
- 取得單項商品：
- http://my-domain/products/17 (GET)
- 新增商品：
- http://my-domain/products (POST)
- 修改商品：
- http://my-domain/products/17 (PUT)
- 刪除商品：
- http://my-domain/products/17 (DELETE)

管理端 URI

- 呈現新增商品的表單：
- http://my-domain/products/add (GET)
- 呈現修改商品的表單：
- http://my-domain/products/17/edit (GET)
- 呈現刪除商品的表單：
- http://my-domain/products/17/delete (GET)

2 前端 AJAX

- XMLHttpRequest
- Fetch API
- Axios（套件）
- jQuery.ajax（jQuery 附屬功能）

2.1 XMLHttpRequest

- XMLHttpRequest 屬性及方法參考： https://developer.mozilla.org/zh-TW/docs/Web/API/XMLHttpRequest
- 事件：onreadystatechange、onload、onerror、onprogress 、onabort 。
- 範例：https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
- 必須在呼叫 open() 方法開啟請求連線之前，就註冊好事件監聽器，否則事件監聽器將不會被觸發。
- 可選擇同步或非同步。
- 上傳進度，使用XMLHttpRequest.upload 物件的事件： load、error、progress 、abort 。

ajax-xhr-01.html

```
function doAjax() {
    var xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function (event) {
        console.log(xhr.readyState, xhr.status);
        console.log(xhr.responseText);
        if(xhr.readyState===4 && xhr.status===200){
            info.innerHTML = xhr.responseText;
        }
    };
    // xhr.open('GET', 'data/sales01.json', true); // 非同步
    xhr.open('GET', 'data/sales01.json', false);   // 同步
    // XMLHttpRequest.open(method, url[, async[, user[, password]]])
    xhr.send();
}
```

2.2 Fetch API

- 參考 https://developer.mozilla.org/zh-TW/docs/Web/API/Fetch_API/Using_Fetch
- 使用 Promise 包裝的 API
- 只有網路錯誤或其他會中斷 request 的情況下，才會發生 reject 。
- 缺點：無法取得上傳或下載的進度過程
- 在傳送少量資料時，是方便的選擇
- 不會主動傳送 cookie，需設定 credentials 為 include。

fetch: 使用 POST 方法

```
const url = 'https://example.com/profile';
const data = {username: 'example'};

fetch(url, {
    method: 'POST',
    body: JSON.stringify(data),
    headers: new Headers({
        'Content-Type': 'application/json'
    })
}).then(res => res.json())
    .catch(error => console.error('Error:', error))
    .then(response => console.log('Success:',
response));
```

fetch: 上傳檔案及表單

```
const formData = new FormData(document.myForm);

fetch('https://example.com/profile/avatar', {
    method: 'PUT',
    body: formData
})
    .then(response => response.json())
    .catch(error => console.error('Error:', error))
    .then(response => console.log('Success:',
response));
```

2.3 Axios

- Axios 工具: https://www.npmjs.com/package/axios

- Axios優點

  1. 方便使用，類似 jQuery 的 ajax 方法
  2. Promise API 包裝
  3. 可以在後端 node.js 中使用
  4. 體積輕量

axios: 上傳檔案及表單

```
axios.post('/try-upload3', formData, {
    headers: {
        'Content-Type': 'multipart/form-data'
    },
    onUploadProgress: function(progressEvent){
        const perc = Math.round(progressEvent.loaded/progressEvent.total*100);
        const t = new Date().toLocaleString();
        const str = `${perc} % : ${t}`;
        console.log(str);
        progress.innerHTML += str+'<br>';
    }

}).then(r=>{
    console.log(r.data);
    console.log(new Date());
});
```

3. 使用 JSON Web Token

- 使用 bcryptjs 套件加密密碼
- 使用 https://www.npmjs.com/package/jsonwebtoken 套件。
- 先決條件：必須在加密的環境中使用，如 HTTPS
- 優點：可在不同的用戶端環境使用，不局限於網站。
- 缺點：需存放在用戶端，由JavaScript 發送，或其他前端技術發送。

用戶端 Demo

```
const jwtData = JSON.parse(localStorage.getItem('jwtTest'));

function send(){
    fetch('/address-book/verify2',{
        headers: {
            Authorization: 'Bearer ' + jwtData.token
        }
    })
        .then(r=>r.json())
        .then(obj=>{
            console.log(obj);
        })
}
```

伺服端 Middleware Demo

```
app.use((req, res, next)=>{
    res.locals.sess = req.session;

    let auth = req.get('Authorization');

    if(auth && auth.indexOf('Bearer ')===0){
        auth = auth.slice(7);
        jwt.verify(auth, process.env.TOKEN_SECRET, function(error, payload){
            if(!error){
                req.bearer = payload;
            }
            next();
        });
    } else {
        next();
    }
})
```

4. API 實作

- 商品 API：
- 1. 列表
- 2. 搜尋
- 3. 單項商品

- 購物車 API：
- 1. 讀取
- 2.加入
- 3.移除
- 4.修改
- 5.清空