

LARAVEL • 2024-05-20

Laravel 11 with Sanctum & Vue3

從前面兩篇 [Laravel Breeze with Vue 3 use Inertia In SSR](#) 與 [Laravel Jetstream with Vue 3 use Inertia In SSR](#)都是 SSR的整合，這篇則是以單純 Laravel 與 Vue 3 的整合，加這 Sanctum 使用 cookie 的認證機制 (Sanctum uses Laravel's cookie-based session authentication to authenticate users from your client.)。

系統環境:

- Washer 11
- PHP 8.2

內容目錄

- 0.1. Laravel Sanctum 認證過程說明
- 1. 建置專案
 - 1.1. Step01: 建立一個新的 Laravel 11 專案
 - 1.2. Step02: 設定資料庫連線
 - 1.3. Step03: 安裝 Sanctum API
 - 1.4. Step04: Sanctum 的設定
 - 1.5. Step 05: Create API Routes
 - 1.6. Step06: Create Controller Files
 - 1.7. Step07: 執行測試 Run Application Server
- 2. PostMan 測試
 - 2.1. 註冊(Register)
 - 2.2. 登入帳號(Login)
 - 2.3. 取得帳號訊息(getLoginUser)
- 3. 安裝 Vue 及 Vite 設定 (Vue Installation and Vite Config)
 - 3.1. 設定前端(Setup Frontend)
 - 3.2. 安裝 Vue Router
 - 3.3. 建立用於登入和註冊的元件。
 - 3.4. Layout (佈局)元件:Dashboard.vue 的版面元件
 - 3.5. 為這些頁面元件新增至路由器(vue-router)
 - 3.6. 建立 Vue 首頁
 - 3.7. web.php 和 api.php 路由檔案中定義路由
 - 3.8. 登入測試
- 4. 程式碼
- 5. 附註

Laravel Sanctum Certification Process Instructions

[Laravel Sanctum](#) 為 SPA (單頁應用程式)、行動應用程式和簡單的基於 Token(憑證) 的 API 提供了一個輕量級的身份驗證系統。 Sanctum 允許應用程式的每個使用者為其帳戶產生多個 API Token(憑證)。 這些Token(憑證)可以被授予指定允許令牌執行哪些操作的能力/範圍。

使用 Laravel 基於 cookie 的會話驗證來對用戶端的使用者進行身份驗證，下面是認證作業過程。

- 從客戶端上的 Sanctum 要求 CSRF cookie，這允許向正常端點 (例如 /login) 發出受 CSRF 保護的請求。

POPULAR

TRENDING

LATEST

LARAVEL • 2024-07-09

[Laravel+vue3+vite 移除console與分析檔案大小](#)

polish road

DATABASE • 2024-06-

28

[使用sqlcmd修改 SQL Server 的 sa 密碼](#)

polish road

SQLSERVER • 2024-06-

28

[sqlcmd 匯出csv 檔&使用批次檔案 \(bat\)](#)

polish road

WINDOWS • 2023-11-19

[Active Directory 操作主機角色移轉與拿取](#)

polish road

TAGS

DNS

linux

the holy

Spring Boot

SSR

CATEGORIES

> Apache

> Database

> Java

> Laravel

> Node.js

> PHP

- 現在，對 API 的任何請求都包含此 cookie，因此使用者在請求(request)的生命週期內都經過身份驗證。
-

construction project

Step01: Create a new Laravel 11 project

```
1 composer create-project laravel/laravel lar
```

Step02: Set up database connection

Laravel 11 版本以後，在檔案 .env 中預設的資料庫為 sqlite，可以不用修改就可以直接使用

```
1 DB_CONNECTION=sqlite
2 # DB_HOST=127.0.0.1
3 # DB_PORT=3306
4 # DB_DATABASE=laravel
5 # DB_USERNAME=root
6 # DB_PASSWORD=
```

If you want to use mysql, you have to modify it as follows

```
1 DB_HOST=127.0.0.1
2 DB_PORT=3306
3 DB_DATABASE=<DATABASE NAME>
4 DB_USERNAME=<DATABASE USERNAME>
5 DB_PASSWORD=<DATABASE PASSWORD>
```

Step03: Install Sanctum API

透過下列的指令安裝 Laravel Sanctum

```
1 php artisan install:api
```

它會改寫 bootstrap\app.php。

```
> bootstrap > app.php > ...
1 <?php
2
3 use Illuminate\Foundation\Application;
4 use Illuminate\Foundation\Configuration\Exceptions;
5 use Illuminate\Foundation\Configuration\Middleware;
6
7 return Application::configure(basePath: dirname(__DIR__))
8     ->withRouting(
9         web: __DIR__.'/../routes/web.php',
10        api: __DIR__.'/../routes/api.php',
11        commands: __DIR__.'/../routes/console.php',
12        health: '/up',
13    )
14    ->withMiddleware(function (Middleware $middleware) {
15        //
16    })
17    ->withExceptions(function (Exceptions $exceptions) {
18        //
19    })
20    ->create();| You, 6 hours ago • init commit
```

並且它也會建立 Sanctum 的配置檔

config/sanctum.php

> VeeValidate

> Quickly

> Vue3

> Windows

> WordPress

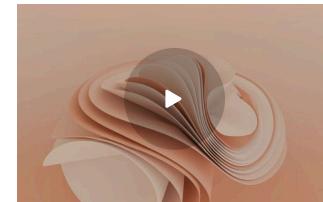
> XAMPP

> 佈景

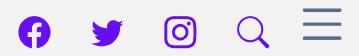
> 偵錯

> 網路技術

VIDEO



PolinWEI Blog



```
4
5     return [
6
7         /*
8         /-----
9         / Stateful Domains
10        /-----
11        /
12        / Requests from the following domains
13        / authentication cookies. Typically, these are local development
14        / and production domains which access the application via
15        /
16        */
17
18     'stateful' => explode(',', env('SANCTUM_STATEFUL_DOMAINS', '%s%s',
19                                     'localhost,localhost:3000,127.0.0.1'))),
20
21     Sanctum::currentApplicationUrlWithToken(),
22 ),
23
24     /*
25     /-----
26     / Sanctum Guards
27     /-----
28     /
29     / This array contains the authentication guards that Sanctum is trying to authenticate a user with. These are able to authenticate the request if they have a valid token that's present on an incoming request.
30     /
31     */
32
33
34     /*
35     /-----
36     / guard' => ['web'],
37     /
38     /*
39     /-----
40     / Expiration Minutes
41     /-----
42     /
43     / This value controls the number of minutes a token is considered expired. This will override the "expires_at" attribute, but first-party applications can still set their own expiration time.
44     /
45     */
46
47     /*
48     /-----
49     / expiration' => null,
50     /
51     /*
52     /-----
53     / Token Prefix
54     /-----
55     /
56     / Sanctum can prefix new tokens in order to help security scanning initiatives maintain a consistent prefix across all tokens. This allows them to notify developers if they commit to a specific prefix.
57     /
58     / See: https://docs.github.com/en/code-security/scanning-for-vulnerabilities/about-scanning-for-vulnerabilities#prefixing-new-tokens
59     /
60     */
61
62     /*
63     /-----
64     / token_prefix' => env('SANCTUM_TOKEN_PREFIX', 'sanctum'),
65     /
66     /*
67     /-----
68     / Sanctum Middleware
69     /-----
70     /
71     / When authenticating your first-party application, you can customize some of the middleware Sanctum uses to handle the authentication request. You may change the middleware, or add your own.
72     /
73     */
74
75     /*
76     /-----
77     / middleware' => [
```

```

  ],
82
83 ];

```

也會加入一條 **api: routes/api.php** 路由

```

1 <?php
2
3 use Illuminate\Http\Request;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('/user', function (Request $request) {
7     return $request→user();
8 })→middleware('auth:sanctum');

```

Step04: Sanctum settings

在此步驟中，我們必須配置三個位置：模型(model)、服務提供者(service provider)和身分驗證設定檔(auth config file)。因此，您只需在這些文件中進行以下更改：在 User 模型中(model)，我們加入了 Sanctum 的 **HasApiTokens** 類別。

app/Models/User.php

```

1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Foundation\Auth\User as Authenticatable;
7 use Illuminate\Notifications\Notifiable;
8 use Laravel\Sanctum\HasApiTokens;
9
10 class User extends Authenticatable
11 {
12     use HasFactory, Notifiable, HasApiTokens;
13
14     /**
15      * The attributes that are mass assignable.
16      *
17      * @var array
18     */
19     protected $fillable = [
20         'name',
21         'email',
22         'password',
23     ];
24
25     /**
26      * The attributes that should be hidden for arrays.
27      *
28      * @var array
29     */
30     protected $hidden = [
31         'password',
32         'remember_token',
33     ];
34
35     /**
36      * Get the attributes that should be casted to native types.
37      *
38      * @return array
39     */
40     protected function casts(): array
41     {
42         return [
43             'email_verified_at' => 'datetime',
44             'password' => 'hashed',
45         ];
46     }

```

Step 05: Create API Routes

在此步驟中，將為登入(login)、註冊(register)和取得用戶資料(user) REST API 建立 API 路由。因此，在該文件中新增一條新路線。

routes/api.php

```

1 <?php
2
3 use App\Http\Controllers\API\AuthController;
4 use Illuminate\Http\Request;
5 use Illuminate\Support\Facades\Route;
6
7 Route::controller(AuthController::class)→
8     Route::post('register', 'register');
9     Route::post('login', 'login');
10 );
11
12 Route::get('/user', function (Request $req)
13 {
    return $request→user();
14 })→middleware('auth:sanctum');
15

```

Step 06: Create Controller Files

在這一步中，我們建立了一個名為 BaseController 和 RegisterController 的新控制器。在 Controllers 資料夾中建立了一個名為「API」的新資料夾，因為將擁有單獨的 API 控制器。

使用指令建立在目錄API下的兩個 controller:

BaseController & AuthController，它會註冊在 Laravel 中，切記要註冊，不然當你執行 **php artisan route:list** 時會出現檔案不存在的錯誤訊息。

```

1 php artisan make:controller API\BaseController
2 php artisan make:controller API\AuthController

```

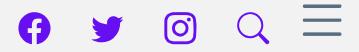
app/Http/Controllers/API/BaseController.php

```

1 <?php
2
3 namespace App\Http\Controllers\API;
4
5 use App\Http\Controllers\Controller;
6 use Illuminate\Http\Request;
7
8 class BaseController extends Controller
9 {
10     /**
11      * success response method.
12      *
13      * @return \Illuminate\Http\Response
14     */
15     public function sendResponse($result,
16     {
17         $response = [
18             'success' => true,
19             'data' => $result,
20             'message' => $message,
21         ];
22
23         return response()→json($response,
24     }
25

```

PolinWEI Blog



```

27     * @return \Illuminate\Http\Response
28     */
29     public function sendError($error, $errorMessages = [])
30     {
31         $response = [
32             'success' => false,
33             'message' => $error,
34         ];
35
36         if(!empty($errorMessages)){
37             $response['data'] = $errorMessages;
38         }
39
40         return response()->json($response, 400);
41     }
42 }
43
44 }
45

```

app/Http/Controllers/API/AuthController.php

```

1  <?php
2
3  namespace App\Http\Controllers\API;
4
5  use Illuminate\Http\Request;
6  use App\Http\Controllers\API\BaseController;
7  use App\Models\User;
8  use Illuminate\Support\Facades\Auth;
9  use Validator;
10 use Illuminate\Http\JsonResponse;
11 class AuthController extends BaseController
12 {
13     /**
14      * Register api
15      *
16      * @return \Illuminate\Http\Response
17      */
18     public function register(Request $request)
19     {
20         $validator = Validator::make($request->all(),
21             [
22                 'name' => 'required',
23                 'email' => 'required|email',
24                 'password' => 'required',
25                 'c_password' => 'required|same:c_password'
26             ]);
27
28         if($validator->fails()){
29             return $this->sendError('Validation error');
30         }
31
32         $input = $request->all();
33         $input['password'] = bcrypt($input['password']);
34         $user = User::create($input);
35         $success['token'] = $user->createToken('auth_token')->plainTextToken;
36         $success['name'] = $user->name;
37
38         return $this->sendResponse($success);
39     }
40
41     /**
42      * Login api
43      *
44      * @return \Illuminate\Http\Response
45      */
46     public function login(Request $request)
47     {
48         if(Auth::attempt(['email' => $request->email,
49                         'password' => $request->password])){
50             $user = Auth::user();
51             $success['token'] = $user->createToken('auth_token')->plainTextToken;
52             $success['name'] = $user->name;
53
54             return $this->sendResponse($success);
55         }
56     }
57 }
58

```

```
58 }
59
```

Step07: Execute test Run Application Server

正常來說，依下列指令已經可以運作，可以使用 postman 來作 API 認證測試。

```
1 // 安裝 laravel 相關套件
2 composer install
3 // 執行 Laravel 服務
4 php artisan serve
5
```

PostMan test

Register

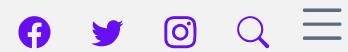
Authorization 不需要設定

In the headers, the application/json format must be accepted

在 Body 要將傳入後台的 request 值填入，此時 AuthController.php 中的 register 函數會新增帳號，並回傳 token 及 name。



PolinWEI Blog



POST ▼ http://127.0.0.1:8000/api/register

Params Authorization Headers (10) **Body** ● Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
<input checked="" type="checkbox"/>	name	Text ▾ admin
<input checked="" type="checkbox"/>	email	Text ▾ admin@example.com
<input checked="" type="checkbox"/>	password	Text ▾ 12345678
<input checked="" type="checkbox"/>	c_password	Text ▾ 12345678
	Key	Text ▾ Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▾

```

1 {
2   "success": true,
3   "data": {
4     "token": "7|GARFz64PAWLifmGmyKvnARYQciUpkCVbJ8k0swIq32e1805c",
5     "name": "admin"
6   },
7   "message": "User register successfully."
8 }
```

Login account(Login)

Authorization does not require setting

HTTP Local Collection / Login

POST ▼ http://127.0.0.1:8000/api/login

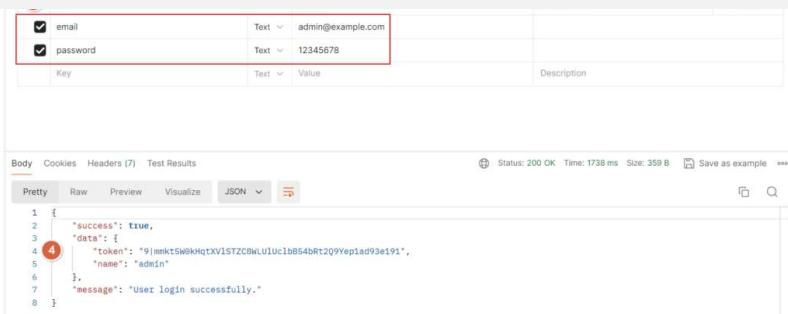
Params Authorization Headers (10) Body ● Scripts Tests Settings

Auth Type

No Auth ▾

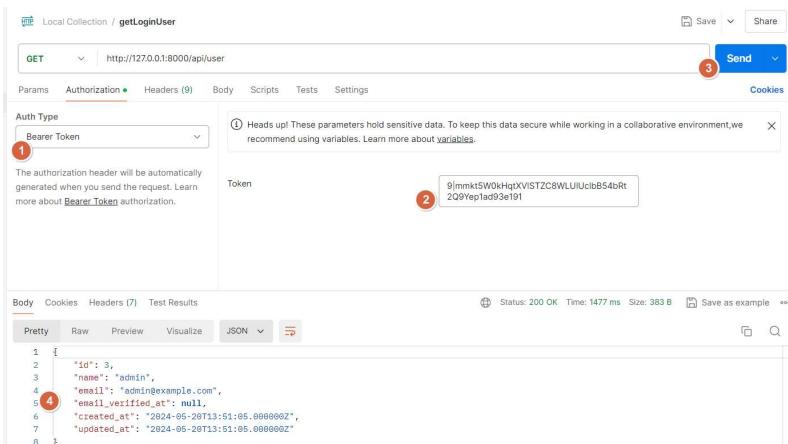
This request does not use

在 Body 中，輸入登入的帳號及密碼，送出 request 後，
會回傳 token 及 name



Get account information (getLoginUser)

在 Authorization 中選擇 Bearer Token , 並填入登入成功後回傳的 token, 送出 request 後, 就可以得到此 token 的資訊了。



Install Vue and Vite Config (Vue Installation and Vite Config)

Setup Frontend

安裝 [Vue Vite](#) plugin, 可以依下列官網的指令

```
1 npm install --save-dev @vitejs/plugin-vue
```

但必需手動修改 vue.config.js

```

1 import { defineConfig } from 'vite';
2 import laravel from 'laravel-vite-plugin';
3 import vue from '@vitejs/plugin-vue';
4
5 export default defineConfig({
6   plugins: [
7     laravel({
8       input: [
9         'resources/sass/app.scss',
10        'resources/js/app.js',
11      ],
12      refresh: true,
13    }),
14    vue({
15      template: {
16        transformAssetUrls: {
17          base: null,
18          includeAbsolute: false
19        },
20      },
21    }),
22  ],
23  resolve: {
24    alias: {
25      vue: 'vue/dist/vue.esm-bundler'
26    }
27  }
28})
  
```

預設情況下， Laravel 外掛提供了一個通用別名 `Aliases` 來幫助您開始運行並輕鬆匯入應用程式的資源：

```
1 {
2     '@' => '/resources/js'
3 }
```

也可以執行下列指令比較方便

```
1 // Setting up the Laravel and Vue Environment
2 composer require laravel/ui // 安裝 laravel,
3 php artisan ui vue           // 安裝 vue 的套
4 npm install
```

當使用 `php artisan ui vue` 產生前端程式碼時，在 `resources/js/components/ExampleComponent.vue` 下產生了一個範例元件。

Install Vue Router

在為登入、註冊和登入後頁面建立其他元件之前，先安

入後頁面映射到路上，讓 Vue Router 知道在哪裡渲染它們。

```
1 npm -D install vue-router
```

建立用於登入和註冊的元件。

在 `resources/js/components` 資料夾名稱中建立一個檔案 `Login.vue`。

```
1 <template>
2     <div class="container h-100">
3         <div class="row h-100 align-items-center justify-content-center">
4             <div class="col-12 col-md-6 offset-md-3">
5                 <div class="card shadow-sm border-0 rounded-0">
6                     <div class="card-body p-4">
7                         <h1 class="text-center mb-4">Login</h1>
8                         <hr/>
9                         <form action="#" method="post">
10                             <div class="mb-3">
11                                 <label for="email" class="form-label">Email address</label>
12                                 <input type="email" class="form-control" id="email" name="email" required="required" />
13                             </div>
14                             <div class="mb-3">
15                                 <label for="password" class="form-label">Password</label>
16                                 <input type="password" class="form-control" id="password" name="password" required="required" />
17                             </div>
18                             <div class="mb-3">
19                                 <label for="remember-me" class="form-label">Remember me</label>
20                                 <input type="checkbox" class="form-check-input" id="remember-me" name="remember" checked="checked" />
21                             </div>
22                             <div class="mb-3">
23                                 <label for="password-confirm" class="form-label">Confirm Password</label>
24                                 <input type="password" class="form-control" id="password-confirm" name="password_confirmation" required="required" />
25                             </div>
26                             <div class="mb-3">
27                                 <button type="submit" class="btn btn-primary w-100" :disabled="loading">
28                                     {{ loading ? 'Please wait...' : 'Login' }}</button>
29                             </div>
30                             <div class="mb-3">
31                                 <label>Don't have an account?</label>
32                                 <a href="#" class="text-decoration-none" style="color: #007bff; font-weight: bold;">Register</a>
33                             </div>
34                         </form>
35                     </div>
36                 </div>
37             </div>
38         </div>
39     </div>
40 
```

```

37     </div>
38   </div>
39 </template>
40
41 <script>
42 import router from '@/router'
43 export default {
44   name:"login",
45   data(){
46     return {
47       auth:{
48         email:"",
49         password:""
50       },
51       validationErrors:{},
52       processing:false
53     }
54   },
55   methods:{
56     async login(){
57       this.processing = true
58       await axios.post('/api/login',
59         // JSON.stringify() : 物件轉成字串
60         localStorage.setItem('user',
61           router.push({name:'dashboard'})
62         ).catch(({response})=>{
63           if(response.status==422){
64             this.validationErrors =
65           }else{
66             this.validationErrors =
67             alert(response.data.message)
68           }
69         }).finally(()=>{
70           this.processing = false
71         })
72       },
73     }
74   }
75 </script>

```

在 resources/js/components 資料夾名稱中建立一個檔案
Register.vue。

```

1 <template>
2   <div class="container h-100">
3     <div class="row h-100 align-items-center">
4       <div class="col-12 col-md-6 offset-md-3">
5         <div class="card shadow-sm border-0 rounded-0">
6           <div class="card-body p-4">
7             <h1 class="text-center mb-4">註冊</h1>
8             <hr/>
9             <form action="#" method="post">
10               <div class="mb-3">
11                 <div class="form-group">
12                   <ul class="list-group list-group-flush">
13                     <li class="list-item">
14                       <ul>
15                         <li></li>
16                       </ul>
17                     </li>
18                   </ul>
19                 </div>
20                 <div class="form-group">
21                   <label for="username">帳號</label>
22                   <input type="text" id="username" class="form-control" />
23                 </div>
24                 <div class="form-group">
25                   <label for="password">密碼</label>
26                   <input type="password" id="password" class="form-control" />
27                 </div>
28                 <div class="form-group">
29                   <label for="confirm_password">確認密碼</label>
30                   <input type="password" id="confirm_password" class="form-control" />
31                 </div>
32               </div>
33             </form>
34           </div>
35         </div>
36       </div>
37     </div>
38   </div>
39 </template>
40
41 <script>
42 import { ref, reactive } from 'vue'
43 import { useRouter } from 'vue-router'
44 import { useAuthStore } from '@/store/auth'
45 import { useUserStore } from '@/store/user'
46 import { useFlash } from '@/composables/useFlash'
47 import { useTitle } from '@/composables/useTitle'
48 import { useMeta } from '@/composables/useMeta'
49 import { useRoute } from 'vue-router'
50
51 const router = useRouter()
52 const authStore = useAuthStore()
53 const userStore = useUserStore()
54 const flash = useFlash()
55 const title = useTitle()
56 const meta = useMeta()
57
58 const form = reactive({
59   username: '',
60   password: '',
61   confirmPassword: ''
62 })
63
64 const errors = reactive({
65   username: '',
66   password: '',
67   confirmPassword: ''
68 })
69
70 const validate = () => {
71   errors.username = !form.username || form.username.length < 3 ? '請輸入帳號' : ''
72   errors.password = !form.password || form.password.length < 6 ? '請輸入密碼' : ''
73   errors.confirmPassword = !form.confirmPassword || form.confirmPassword !== form.password ? '請確認密碼' : ''
74 }
75
76 const register = async () => {
77   validate()
78   if (!Object.values(errors).includes('')) {
79     const response = await authStore.register(form)
80     if (response?.status === 200) {
81       userStore.setUser(response.data)
82       flash.success('註冊成功')
83       router.push({ name: 'login' })
84     } else {
85       flash.error('註冊失敗')
86     }
87   }
88 }
89
90 </script>
91
92 <script setup>
93
94 import { computed } from 'vue'
95 import { useRoute } from 'vue-router'
96
97 const route = useRoute()
98
99 </script>

```

```

35             <button type="button" @click="register">
36                 {{ processing ? 'Processing' : 'Register' }}
37             </button>
38         </div>
39         <div class="col-md-6" style="text-align: center;">
40             <label>Already have an account? <a href="#">Log In</a></label>
41         </div>
42     </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 </template>
48
49 <script>
50 import router from '@/router'
51 export default {
52     name:'register',
53     data(){
54         return {
55             user:{
56                 name:"",
57                 email:"",
58                 password:"",
59                 c_password:""
60             },
61             validationErrors:{},
62             processing:false
63         }
64     },
65     methods:{
66         async register(){
67             this.processing = true
68             await axios.post('/api/register')
69             localStorage.clear()
70             this.validationErrors = {}
71             console.log(response.data)
72             localStorage.setItem('user', JSON.stringify(response.data))
73             router.push({name:'dashboard'})
74         }.catch(({response})=>{
75             if(response.status==422){
76                 this.validationErrors = response.data.errors
77             }else{
78                 this.validationErrors = {}
79                 alert(response.data.message)
80             }
81         }).finally(()=>{
82             this.processing = false
83         })
84     }
85 }
86 </script>
87 
```

Layout (佈局)元件:Dashboard.vue 的版面元件

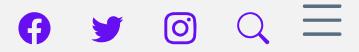
為所有經過身份驗證的頁面建立 Layout (佈局)元件。因此，不需要在所有頁面元件中新增頁首、頁尾和任何其他元件，因此這裡建立了一個名為 Dashboard.vue 的版面元件。在元件中，新增頁首、頁尾和路由器視圖，以便每個元件都會在此路由器視圖中呈現。

[resources/js/components/layouts/Default.vue](#)

```

1 <template>
2     <div>
3         <nav class="navbar navbar-expand-lg">
4             <div class="container-fluid">
5                 <a class="navbar-brand" href="#">PolinWEI Blog</a>
6             </div>
7         </nav>
8         <div class="row" style="margin-top: 20px;">
9             <div class="col-md-3" style="background-color: #f2f2f2; padding: 10px; border-radius: 5px; min-height: 400px;">
10                <h3>Dashboard</h3>
11                <p>This is the dashboard page for PolinWEI Blog. It contains links to various sections of the blog, such as Posts, Categories, and Tags. It also includes a sidebar with navigation menus and search functionality.</p>
12            </div>
13            <div class="col-md-9" style="background-color: #f2f2f2; padding: 10px; border-radius: 5px; min-height: 400px;">
14                <h3>Recent Posts</h3>
15                <ul style="list-style-type: none; padding-left: 0;">
16                    <li>Post 1</li>
17                    <li>Post 2</li>
18                    <li>Post 3</li>
19                </ul>
20            </div>
21        </div>
22    </div>
23 </template>
24
25 <script>
26 import { ref, onMounted } from 'vue'
27 import { useRoute } from 'vue-router'
28
29 export default {
30     setup() {
31         const route = useRoute()
32         const title = ref(route.meta.title)
33
34         onMounted(() => {
35             document.title = title.value
36         })
37     }
38 }
39 </script>
40
41 <style>
42 .row { display: flex; justify-content: space-between; align-items: flex-start; }
43 .col-md-3 { width: 30%; }
44 .col-md-9 { width: 70%; }
45 </style>
46 
```

PolinWEI Blog



```

1   <div class="collapse navbar-collapse">
2     <ul class="navbar-nav">
3       <li class="nav-item">
4         <router-link :to="user ? '/profile' : '/login'">
5           </li>
6         </ul>
7         <div class="d-flex">
8           <ul class="navbar-nav">
9             <li class="nav-item">
10               <a class="nav-link" href="#">
11                 {{ user ? 'Logout' : 'Login' }} </a>
12               <div class="dropdown">
13                 <a class="dropdown-item" href="#">Profile</a>
14               </div>
15             </li>
16           </ul>
17         </div>
18       </div>
19     </div>
20   </nav>
21   <main class="mt-3">
22     <router-view></router-view>
23   </main>
24 </div>
25 </template>
26
27 <script>
28 export default {
29   name:"default-layout",
30   data(){
31     return {
32       user:[]
33     }
34   },
35   mounted() {
36     this.user = JSON.parse(localStorage.getItem("user"));
37   },
38   methods:{
39     logout(){
40       setTimeout(() => {
41         localStorage.clear();
42         this.$router.push({name:"login"});
43       }, 150);
44     }
45   }
46 }
47 </script>
48 
```

resources/js/components/Dashboard.vue

```

1 <template>
2   <div class="container">
3     <div class="row">
4       <div class="col-12">
5         <div class="card shadow-sm border-0 rounded-0">
6           <div class="card-header p-2 border-bottom-0">
7             <h3>Dashboard</h3>
8           </div>
9           <div class="card-body p-2">
10            <p class="mb-0">Your dashboard content here</p>
11            <p class="mb-0">Another dashboard item</p>
12          </div>
13        </div>
14      </div>
15    </div>
16  </div>
17 </template>
18
19 <script>
20 export default {
21   name:"dashboard",
22   data(){
23     return {
24       user:[]
25     }
26   },
27   mounted() {
28     this.user = JSON.parse(localStorage.getItem("user"));
29   },
30   methods:{
31     logout(){
32       setTimeout(() => {
33         localStorage.clear();
34         this.$router.push({name:"login"});
35       }, 150);
36     }
37   }
38 }
39 </script>
40 
```

```

26      }
27    },
28    async mounted() {
29      this.user = JSON.parse(localStorage.getItem('user'))
30      let token = this.user?.data.token;
31      const authHeader = {
32        'Authorization': 'Bearer ' + token,
33        'X-REQUEST-TYPE': 'axios'
34      }
35      let config = {headers:authHeader}
36      await axios.get('/api/user',config)
37        this.userData=data;
38        router.push({name:'dashboard'})
39      }).catch(({response})=>{
40        if(response){
41          this.validationErrors=response.data.error;
42          alert(response?.data.message)
43        }
44      })
45    }
46  }
47 </script>
48

```

為這些頁面元件新增至路由器(vue-router)

新增檔案 [resources/js/router/index.js](#)

```

1 import { createWebHistory, createRouter } from 'vue-router'
2
3 /* Guest Component */
4 const Login = () => import('@/components/Login.vue')
5 const Register = () => import('@/components/Register.vue')
6 /* Guest Component */
7
8 /* Layouts */
9 const DashboardLayout = () => import('@/layouts/DashboardLayout.vue')
10 /* Layouts */
11
12 /* Authenticated Component */
13 const Dashboard = () => import('@/components/Dashboard.vue')
14 /* Authenticated Component */
15
16
17
18 const routes = [
19   {
20     name: "login",
21     path: "/login",
22     component: Login,
23     meta: {
24       middleware: "guest",
25       title: `Login`
26     }
27   },
28   {
29     name: "register",
30     path: "/register",
31     component: Register,
32     meta: {
33       middleware: "guest",
34       title: `Register`
35     }
36   },
37   {
38     path: "/",
39     component: DashboardLayout,
40     meta: {
41       middleware: "auth"
42     }
43   }
44 ]

```

```

46     name: 'dashboard',
47     path: '/',
48     component: Dashboard,
49     meta: {
50       title: `Dashboard`
51     }
52   ]
53 }
54 ]
55
56 const router = createRouter({
57   history: createWebHistory(),
58   routes, // short for `routes: routes`
59 })
60
61 export default router
62

```

將 router 加入 `resources/js/app.js`

```

1 import './bootstrap';
2 import '../sass/app.scss'
3 import { createApp } from 'vue';
4
5 const app = createApp({});
6
7 import ExampleComponent from './components'
8 app.component('example-component', Example
9
10 import Router from '@/router'
11 app.use(Router)
12
13 app.mount('#app');

```

建立 Vue 首頁

Add `resources/views/appHome.blade.php`

```

1 <!DOCTYPE html>
2 <html lang="{{ str_replace('_', '-', app()
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="wid
6     <title>SPA Authentication using La
7     <!-- Fonts -->
8     <link href="https://fonts.bunny.ne
9       @vite(['resources/js/app.js'])
10    </head>
11    <body>
12      <div id="app">
13        <router-view></router-view>
14      </div>
15    </body>
16  </html>

```

義路由

現在在 `web.php` 和 `api.php` 路由檔案中定義路由。轉到
路資料夾並打開 `web.php` 檔案並更新以下路由：

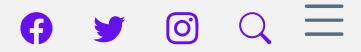
`routes/web.php`

```

1 <?php
2
3 use Illuminate\Support\Facades\Route;

```

PolinWEI Blog



```

        return view('welcome'),
    });

10 Route::get('{any}', function () {
11     return view('appHome');
12 })->where('any', '.*');

13 Auth::routes();

15

```

routes/api.php

```

1 <?php
2
3 use App\Http\Controllers\API\AuthController;
4 use Illuminate\Http\Request;
5 use Illuminate\Support\Facades\Route;
6
7 Route::controller(AuthController::class)->
8     Route::post('register', 'register');
9     Route::post('login', 'login');
10 );
11
12 Route::get('/user', function (Request $request) {
13     return $request->user();
14 })->middleware('auth:sanctum');
15

```

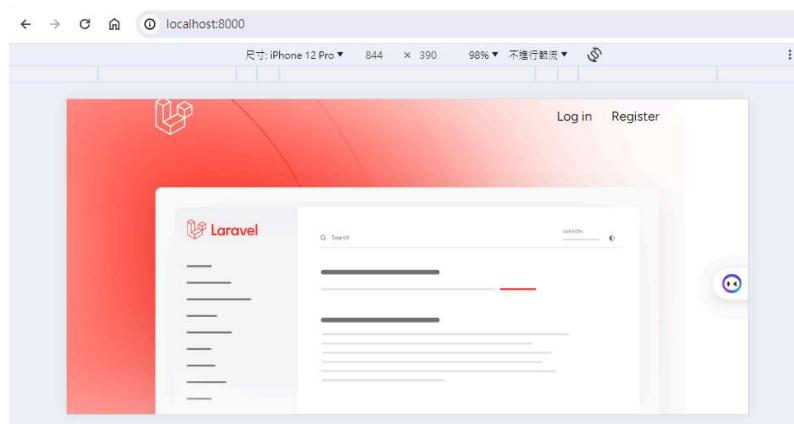
```

1 // 安裝 laravel 相關套件
2 composer install
3 // 執行 Laravel 服務
4 php artisan serve
5
6 // 安裝 npm 套件
7 npm install
8 // 執行 vite
9 npm run dev
10 或
11 npm run build

```

登入測試

<http://localhost:8000/>



點選 Register

Register

Name
Enter name

Email
Enter Email

Password
Enter Password

Confirm Password
Enter Password

Register

Already have an account? [Login Now!](#)

點選 Login

Login

Email

Password

Login

Don't have an account? [Register Now!](#)

登入後的首頁

PolinWEI Blog

Dashboard

You are logged in as **admin**

You Detail is below

```
{ "id": 3, "name": "admin", "email": "admin@example.com", "email_verified_at": null, "created_at": "2024-05-20T13:51:05.000000Z", "updated_at": "2024-05-20T13:51:05.000000Z" }
```

程式碼

程式碼: <https://github.com/polinwei/laravel-vue3>

附註

參考: [CORS and Cookies](#)

```
1 php artisan config:publish cors
```

參考:

[Laravel 11 REST API Authentication using Sanctum Tutorial](#)

[SPA Authentication using Laravel 9 Sanctum, Vue 3 and Vite](#)

« [Laravel Jetstream with Vue 3 use Inertia In SSR 使用sqlcmd修改SQL Server 的 sa 密碼](#) »

發佈留言

發佈留言必須填寫的電子郵件地址不會公開。 必填欄位
標示為 *

顯示名稱 *

電子郵件地址 *

個人網站網址



在瀏覽器中儲存顯示名稱、電子郵件地址及個人網站網址，以供下次發佈留言時使用。

[發佈留言](#)

About PolinWEI

這裡的文章都是工作上的記錄，是為了日後有問題時可以隨時查詢。在工作之餘到處去旅遊，將各地的景點放在每篇文章的精選圖片，以稍解工作上的煩悶。

[Learn More](#)

Categories

- > Apache
- > Database
- > Java
- > Laravel
- > Node.js
- > PHP
- > SQL Server
- > Ubuntu
- > VeeValidate
- > Quickly
- > Vue3
- > Windows
- > WordPress
- > XAMPP
- > 佈景
- > 偵錯
- > 網路技術

Recent Posts

PolinWEI Blog



SQLSERVER • 2024-06-28

sqlcmd export csv file & use batch file (bat)

DATABASE • 2024-06-28

Use sqlcmd to change the sa password of SQL Server

© Copyright ZenBlog. All Rights Reserved

Designed by BootstrapMade 程式撰寫 By PolinWEI

