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

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

```
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

```
DB_CONNECTION=sqlite
# DB_HOST=127.0.0.1
# DB_PORT=3306
# DB_DATABASE=laravel
# DB_USERNAME=root
# DB_PASSWORD=
```

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

```
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=<DATABASE NAME>
DB_USERNAME=<DATABASE USERNAME>
DB_PASSWORD=<DATABASE PASSWORD>
```

Step03: Install Sanctum API

透過下列的指令安裝 Laravel Sanctum

```
php artisan install:api
```

它會改寫 **bootstrap\app.php**。

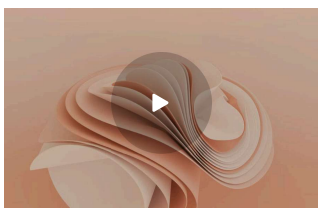
```
bootstrap > app.php > ...
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware) {
        //
    })
    ->withExceptions(function (Exceptions $exceptions) {
        //
    })->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



```php
return [

    /*
    |--------------------------------------
    | Stateful Domains
    |--------------------------------------
    |
    | Requests from the following domains
    | authentication cookies. Typically, t
    | and production domains which access
    |
    */

    'stateful' => explode(',', env('SANCTU
        '%s%s',
        'localhost,localhost:3000,127.0.0.
        Sanctum::currentApplicationUrlWith
    ))),

    /*
    |--------------------------------------
    | Sanctum Guards
    |--------------------------------------
    |
    | This array contains the authenticati
    | Sanctum is trying to authenticate a
    | are able to authenticate the request
    | token that's present on an incoming
    |
    */

    'guard' => ['web'],

    /*
    |--------------------------------------
    | Expiration Minutes
    |--------------------------------------
    |
    | This value controls the number of mi
    | considered expired. This will overri
    | "expires_at" attribute, but first-pa
    |
    */

    'expiration' => null,

    /*
    |--------------------------------------
    | Token Prefix
    |--------------------------------------
    |
    | Sanctum can prefix new tokens in ord
    | security scanning initiatives mainta
    | that notify developers if they commi
    |
    | See: https://docs.github.com/en/code
    |
    */

    'token_prefix' => env('SANCTUM_TOKEN_P

    /*
    |--------------------------------------
    | Sanctum Middleware
    |--------------------------------------
    |
    | When authenticating your first-party
    | customize some of the middleware San
    | request. You may change the middlewa
    |
    */

    'middleware' => [
```

```
        ],
    ];
```

也會加入一條 **api: routes/api.php** 路由

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::get('/user', function (Request $requ
    return $request->user();
})->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

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories
use Illuminate\Foundation\Auth\User as Aut
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasFactory, Notifiable, HasApiToke

    /**
     * The attributes that are mass assign
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidde
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be c
     *
     * @return array
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'dateti
            'password' => 'hashed',
        ];
```

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

```php
<?php

use App\Http\Controllers\API\AuthControlle
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::controller(AuthController::class)->
    Route::post('register', 'register');
    Route::post('login', 'login');
});

Route::get('/user', function (Request $req
    return $request->user();
})->middleware('auth:sanctum');

```

Step06: 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 contorller: 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。

```
php artisan make:controller API\BaseControl
php artisan make:controller API\AuthControl
```

app/Http/Controllers/API/BaseController.php

```php
<?php

namespace App\Http\Controllers\API;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class BaseController extends Controller
{
        /**
     * success response method.
     *
     * @return \Illuminate\Http\Response
     */
    public function sendResponse($result,
    {
        $response = [
            'success' => true,
            'data'    => $result,
            'message' => $message,
        ];

        return response()->json($response,
    }

```

```
     * @return \Illuminate\Http\Response
     */
    public function sendError($error, $err
    {
        $response = [
            'success' => false,
            'message' => $error,
        ];

        if(!empty($errorMessages)){
            $response['data'] = $errorMess
        }

        return response()->json($response,
    }
}
```

app/Http/Controllers/API/AuthController.php

```
<?php

namespace App\Http\Controllers\API;

use Illuminate\Http\Request;
use App\Http\Controllers\API\BaseControlle
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Validator;
use Illuminate\Http\JsonResponse;
class AuthController extends BaseControlle
{
    /**
     * Register api
     *
     * @return \Illuminate\Http\Response
     */
    public function register(Request $requ
    {
        $validator = Validator::make($requ
            'name' => 'required',
            'email' => 'required|email',
            'password' => 'required',
            'c_password' => 'required|same
        ]);

        if($validator->fails()){
            return $this->sendError('Valid
        }

        $input = $request->all();
        $input['password'] = bcrypt($input
        $user = User::create($input);
        $success['token'] =  $user->create
        $success['name'] =  $user->name;

        return $this->sendResponse($succes
    }

    /**
     * Login api
     *
     * @return \Illuminate\Http\Response
     */
    public function login(Request $request
    {
        if(Auth::attempt(['email' => $requ
            $user = Auth::user();
            $success['token'] =  $user->cr
            $success['name'] =  $user->nam

            return $this->sendResponse($su
        }
```

```
58  }
59
```

Step07: Execute test Run Application Server

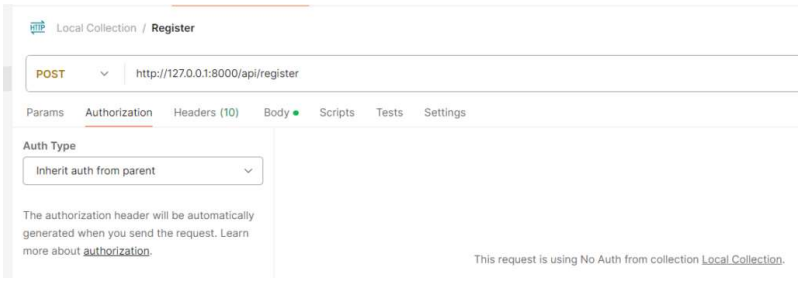
正常來說，依下列指令已經可以運作，可以使用 postman 來作 API 認證測試。

```
// 安裝 laravel 相關套件
composer install
// 執行 Laravel 服務
php artisan serve

```

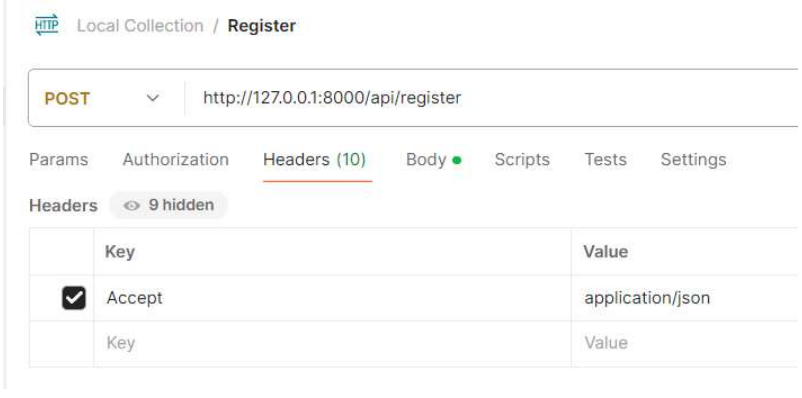
PostMan test

Register

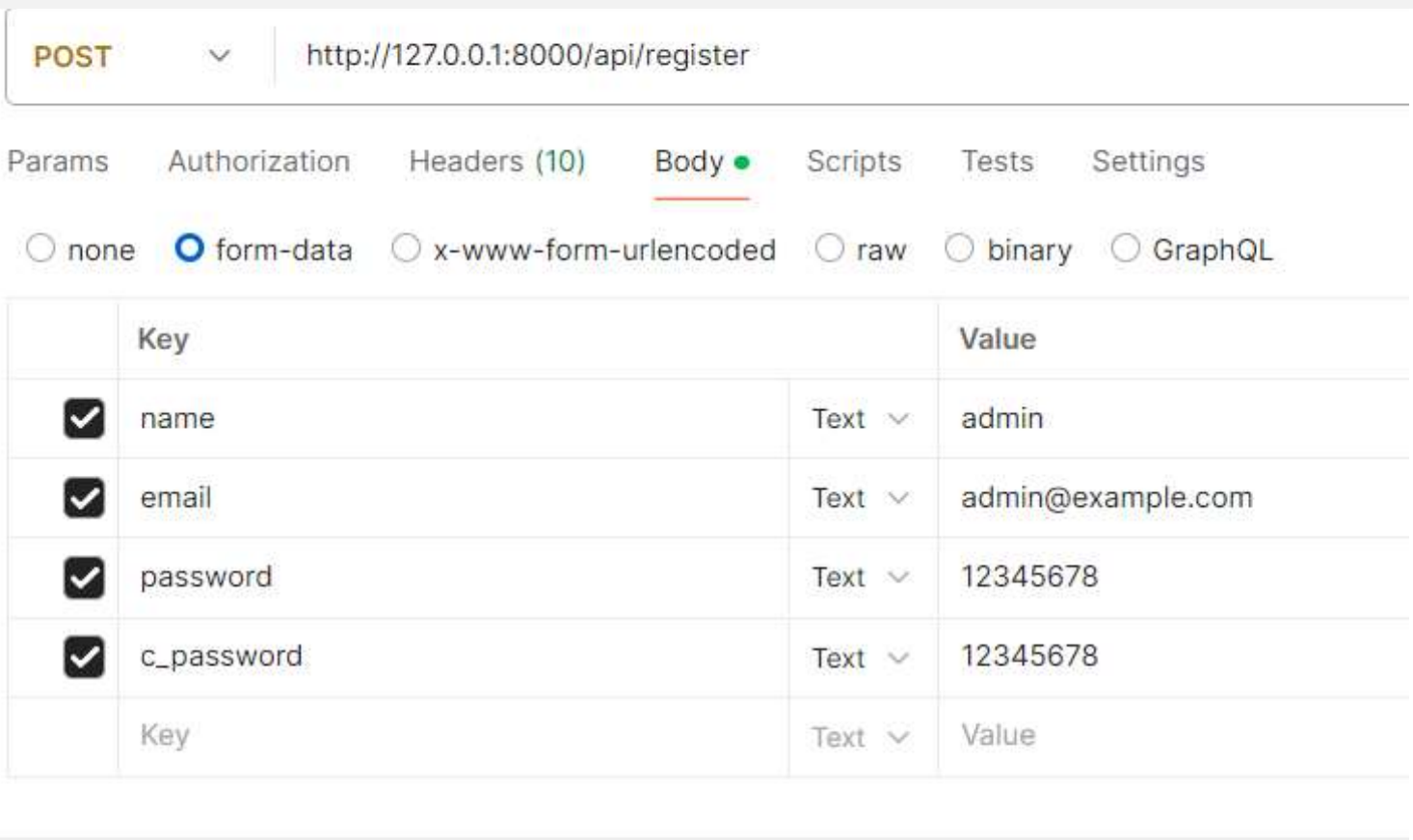
Authorization 不需要設定



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

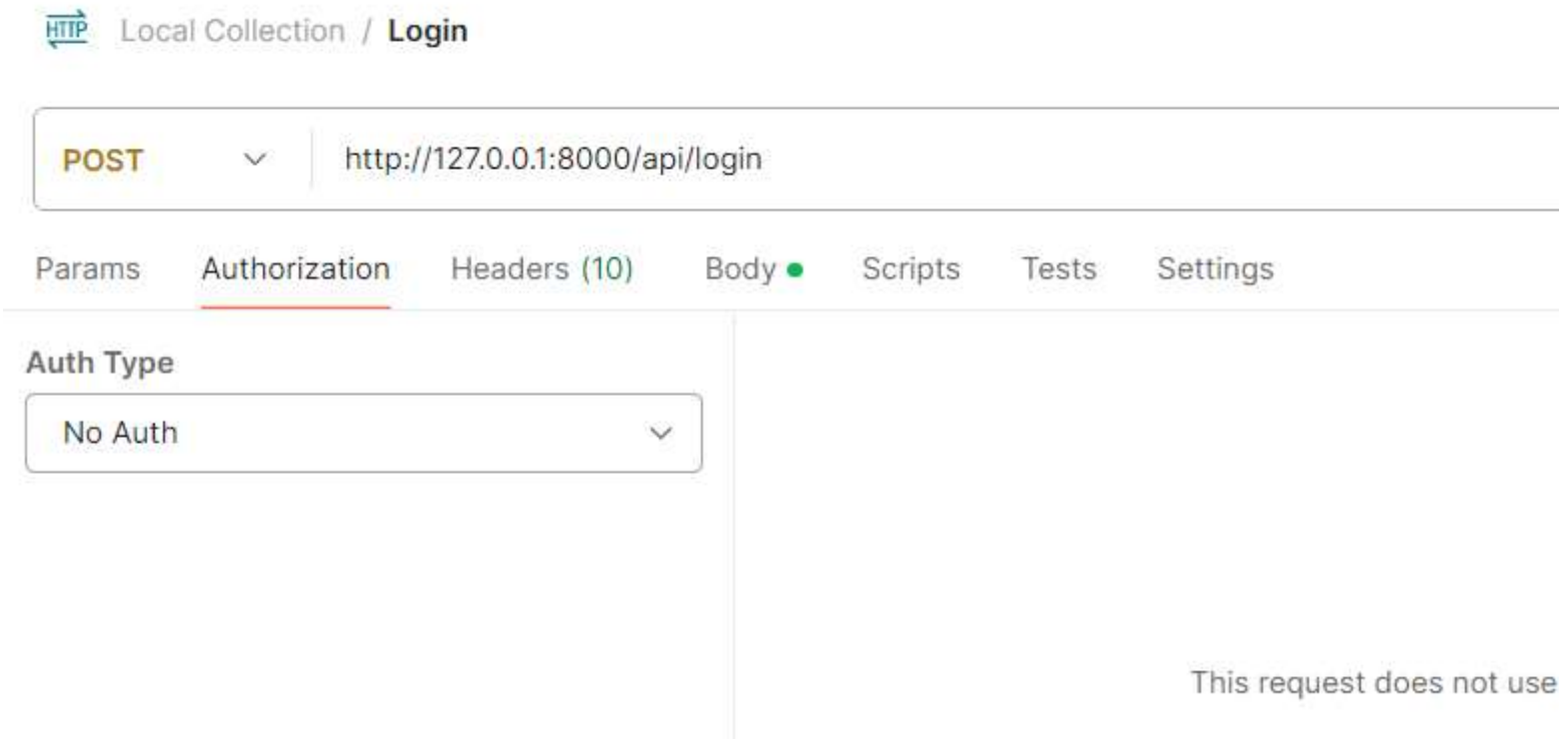


在 Boby 要將傳入後台的 request 值填入，此時 AuthController.php 中的 register 函數會新增帳號，並回傳 token 及 name。

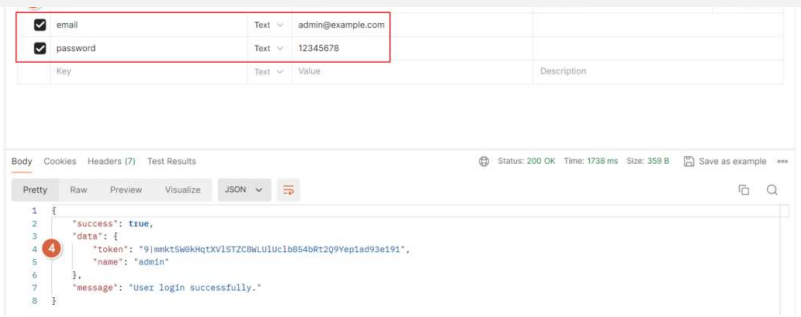


Login account(Login)

Authorization does not require setting

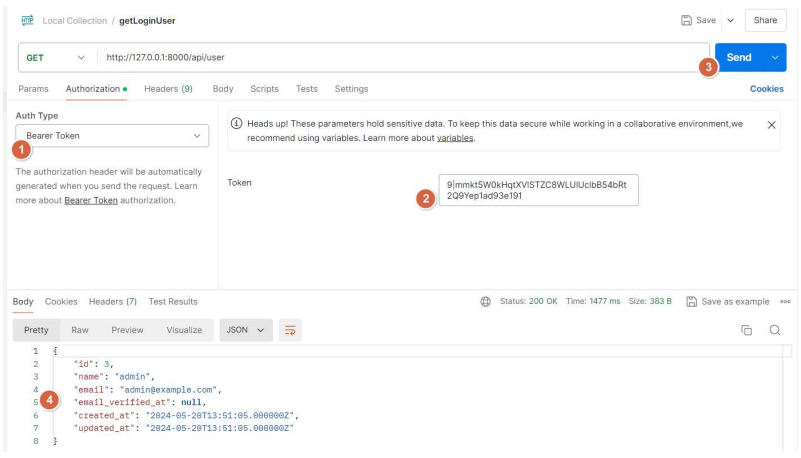


在 Body 中，輸入登入的帳號及密碼，送出 request 後，
會回傳 token 及 name



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 reuqest 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

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

但必需手動修改 vue.config.js

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';
import vue from '@vitejs/plugin-vue';

export default defineConfig({
    plugins: [
        laravel({
            input: [
                'resources/sass/app.scss',
                'resources/js/app.js',
            ],
            refresh: true,
        }),
        vue({
            template: {
                transformAssetUrls: {
                    base: null,
                    includeAbsolute: false
                },
            },
        }),
    ],
    resolve: {
        alias: {
            vue: 'vue/dist/vue.esm-bundler
```

```
```

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：

```
{
    '@' ⇒ '/resources/js'
}
```

也可以執行下列指令比較方便

```
// Setting up the Laravel and Vue Environme
composer require laravel/ui // 安裝 laravel/
php artisan ui vue          // 安裝 vue 的套
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

入後頁面映射到路由上，讓 Vue Router 知道在哪裡渲染它們。

```
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**。

```
<template>
    <div class="container h-100">
        <div class="row h-100 align-items-
            <div class="col-12 col-md-6 of
                <div class="card shadow sm
                    <div class="card-body"
                        <h1 class="text-ce
                        <hr/>
                    <hr/>
                    <form action="java
                        <div class="co
                            <div class
                                <ul cl
                                    <l
                                </ul>
                            </div>
                        </div>
                        <div class="fo
                            <label for
                            <input typ
                        </div>
                        <div class="fo
                            <label for
                            <input typ
                        </div>
                        <div class="co
                            <button ty
                                {{ pro
                            </button>
                        </div>
                        <div class="co
                            <label>Don
                        </div>
                    </form>
```

```
        </div>
    </template>

    <script>
    import router from '@/router'
    export default {
        name:"login",
        data(){
            return {
                auth:{
                    email:"",
                    password:""
                },
                validationErrors:{},
                processing:false
            }
        },
        methods:{
            async login(){
                this.processing = true
                await axios.post('/api/login',
                    // JSON.stringify() : 物件變
                    localStorage.setItem('user
                    router.push({name:'dashboa
                }).catch(({response})=>{
                    if(response.status===422){
                        this.validationErrors
                    }else{
                        this.validationErrors
                        alert(response.data.me
                    }
                }).finally(()=>{
                    this.processing = false
                })
            },
        }
    }
    </script>
```

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

```
<template>
    <div class="container h-100">
        <div class="row h-100 align-items-
            <div class="col-12 col-md-6 of
                <div class="card shadow sm
                    <div class="card-body"
                        <h1 class="text-ce
                        <hr/>
                        <form action="java
                            <div class="co
                                <div class
                                    <ul cl
                                        <l
                                    </ul>
                                </div>
                            </div>
                            <div class="fo
                                <label for
                                <input typ
                            </div>
                            <div class="fo
                                <label for
                                <input typ
                            </div>
                            <div class="fo
                                <label for
                                <input typ
                            </div>
                            <div class="fo
                                <label for
```

```
                                    <button ty
                                        {{ pro
                                    </button>
                                </div>
                                <div class="co
                                    <label>Alr
                                </div>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </template>

        <script>
        import router from '@/router'
        export default {
            name:'register',
            data(){
                return {
                    user:{
                        name:"",
                        email:"",
                        password:"",
                        c_password:""
                    },
                    validationErrors:{},
                    processing:false
                }
            },
            methods:{
                async register(){
                    this.processing = true
                    await axios.post('/api/registe
                        localStorage.clear()
                        this.validationErrors = {}
                        console.log(response.data)
                        localStorage.setItem('user
                        router.push({name:'dashboa
                    }).catch(({response})=>{
                        if(response.status===422){
                            this.validationErrors
                        }else{
                            this.validationErrors
                            alert(response.data.me
                        }
                    }).finally(()=>{
                        this.processing = false
                    })
                }
            }
        }
        </script>
```

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

```
<template>
    <div>
        <nav class="navbar navbar-expand-l
            <div class="container-fluid">
                <a class="navbar-brand" hr
```

```
                <div class="collapse navba
                    <ul class="navbar-nav
                        <li class="nav-ite
                            <router-link :
                        </li>
                    </ul>
                    <div class="d-flex">
                        <ul class="navbar-
                            <li class="nav
                                <a class="
                                    {{ use
                                </a>
                                <div class
                                    <a cla
                                </div>
                            </li>
                        </ul>
                    </div>
                </div>
            </div>
        </nav>
        <main class="mt-3">
            <router-view></router-view>
        </main>
    </div>
</template>

<script>
export default {
    name:"default-layout",
    data(){
        return {
            user:[]
        }
    },
    mounted() {
        this.user = JSON.parse(localStorag
    },
    methods:{
        logout(){
            setTimeout(() => {
                localStorage.clear()
                this.$router.push({name:"l
            }, 150);

        }
    }
}
</script>
```

resources/js/components/Dashboard.vue

```
<template>
    <div class="container">
        <div class="row">
            <div class="col-12">
                <div class="card shadow-sm
                    <div class="card-heade
                        <h3>Dashboard</h3>
                    </div>
                    <div class="card-body"
                        <p class="mb-0">Yo
                        <p class="mb-0">Yo
                    </div>
                </div>
            </div>
        </div>
    </div>
</template>

<script>
export default {
    name:"dashboard",
```

```
            }
        },
        async mounted() {
            this.user = JSON.parse(localStorag
            let token = this.user?.data.token;
            const authHeader = {
                'Authorization': 'Bearer ' + t
                'X-REQUEST-TYPE': 'axios'
            }
            let config = {headers:authHeader}
            await axios.get('/api/user',config
                    this.userData=data;
                    router.push({name:'dashboa
                }).catch(({response})=>{
                    if(response){
                        this.validationErrors
                        alert(response?.data.m
                    }
                })
        }
    }
    </script>
```

為這些頁面元件新增至路由器(vue-router)

新增檔案 `resources/js/router/index.js`

```
import { createWebHistory, createRouter }


/* Guest Component */
const Login = () => import('@/components/L
const Register = () => import('@/component
/* Guest Component */

/* Layouts */
const DahboardLayout = () => import('@/com
/* Layouts */

/* Authenticated Component */
const Dashboard = () => import('@/componen
/* Authenticated Component */


const routes = [
    {
        name: "login",
        path: "/login",
        component: Login,
        meta: {
            middleware: "guest",
            title: `Login`
        }
    },
    {
        name: "register",
        path: "/register",
        component: Register,
        meta: {
            middleware: "guest",
            title: `Register`
        }
    },
    {
        path: "/",
        component: DahboardLayout,
        meta: {
            middleware: "auth"
```

```
                name: 'dashboard',
                path: '/',
                component: Dashboard,
                meta: {
                    title: `Dashboard`
                }
            }
        ]
    }
]

const router = createRouter({
    history: createWebHistory(),
    routes, // short for `routes: routes`
})

export default router
```

將 router 加入 `resources/js/app.js`

```
import './bootstrap';
import '../sass/app.scss'
import { createApp } from 'vue';

const app = createApp({});

import ExampleComponent from './components
app.component('example-component', Example

import Router from '@/router'
app.use(Router)

app.mount('#app');
```

建立 Vue 首頁

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

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="wid
        <title>SPA Authentication using La
        <!-- Fonts -->
        <link href="https://fonts.bunny.ne
        @vite(['resources/js/app.js'])
    </head>
    <body>
        <div id="app">
            <router-view></router-view>
        </div>
    </body>
</html>
```

義路由

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

routes/web.php

```
<?php

use Illuminate\Support\Facades\Route;
```

```php
});

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

Auth::routes();
```

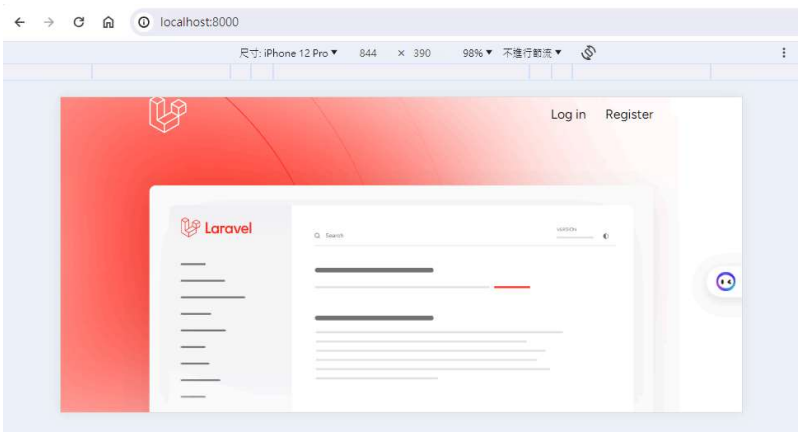
routes/api.php

```php
<?php

use App\Http\Controllers\API\AuthControlle
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::controller(AuthController::class)->
    Route::post('register', 'register');
    Route::post('login', 'login');
});

Route::get('/user', function (Request $req
    return $request->user();
})->middleware('auth:sanctum');
```

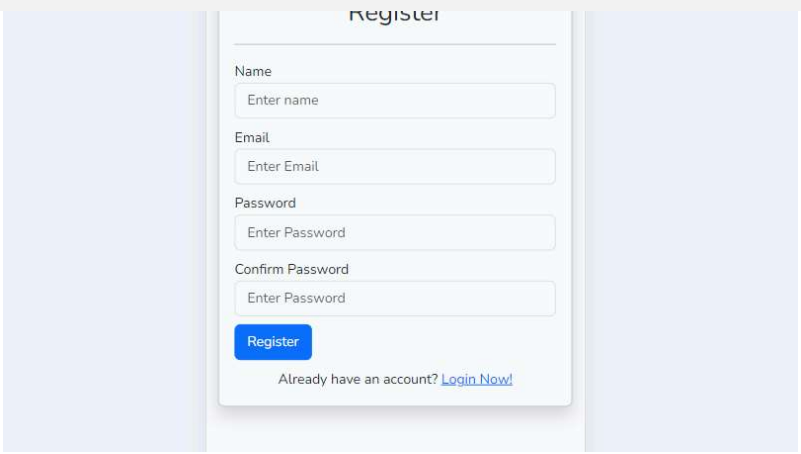
```bash
// 安裝 laravel 相關套件
composer install
// 執行 Laravel 服務
php artisan serve

// 安裝 npm 套件
npm install
// 執行 vite
npm run dev
或
npm run build
```

登入測試

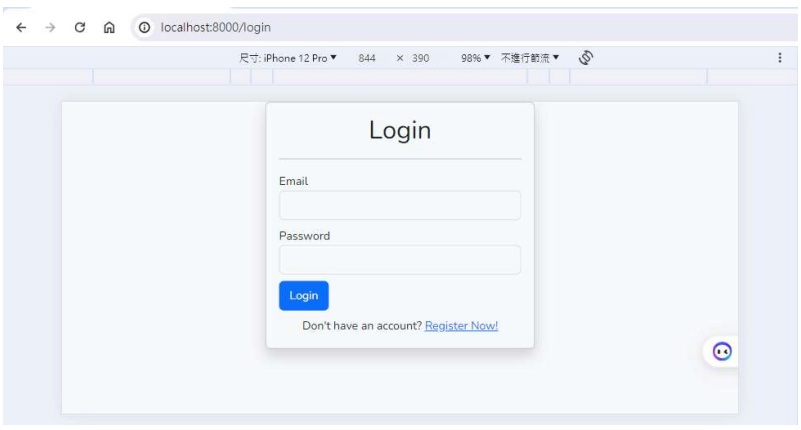
http://localhost:8000/



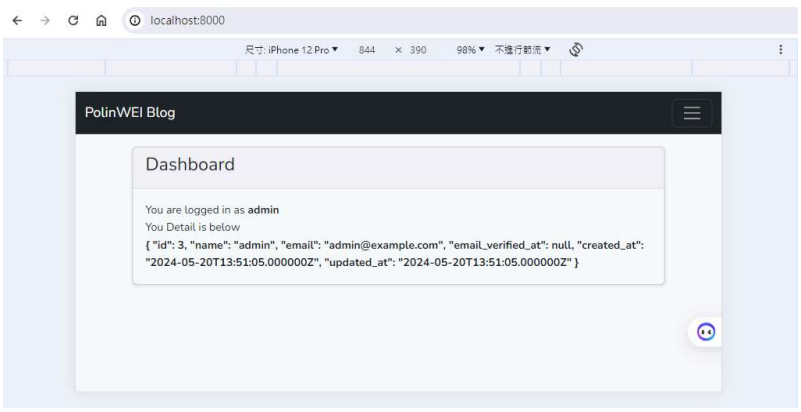
點選 Register



點選 Login



登入後的首頁



程式碼

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

附註

參考: CORS and Cookies

```
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

Recent Posts

LARAVEL • 2024-07-09

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