

Laravel

- [Laravel 建立專案](#)
- [Laravel 常見問題: Specified key was too long](#)
- [Laravel 解決在 CentOS 7 下 log 檔無法寫入的問題](#)
- [Laravel jwt登入驗證](#)
- [Laravel migration 可使用的欄位類型](#)
- [Laravel queue 使用supervisor 實現多執行序](#)
- [Laravel Repository](#)
- [Laravel_解決CORS錯誤](#)
- [Laravel_log_permission](#)
- [Laravel_Queue](#)
- [Laravel_Request](#)
- [Laravel_Validation](#)
- [【Larvel】 redirect\(\) 與 redirect\(\)->intended\(\)](#)
- [Laravel 8 + Vue 3 專案建置 \(PHP 7.4 支援\)](#)

Laravel 建立專案

先安裝composer ，在切換到放置專案的目錄

```
composer global require laravel/installer
```

```
laravel new example-app
```

```
cd example-app
```

```
php artisan serve
```

修改參數

```
//config/app.php
```

```
'timezone' => 'Asia/Taipei',
```

```
'locale' => 'zh_TW',
```

```
'faker_locale' => 'zh_TW',
```

加上Vue

[laravel8 + vue3.0 + element-plus 搭建 | IT人 \(iter01.com\)](#)

[Laravel SPA with Vue 3, Auth \(Sanctum\), CRUD Example - Shouts](#)

Laravel 常見問題: Specified key was too long

Laravel 在 5.4 版之後為了支援 emoji，因此將資料編碼改為 utf8mb4。由於 utf8mb4 的儲存空間需求膨脹了4倍，導致預設長度無法正常寫入資料庫。

這個問題會在 MySQL 5.7.6 以下與 MariaDB 的環境中出現以下錯誤

```
[PDOException]
SQLSTATE[42000]: Syntax error or access violation: 1071 Specified key was too long; max key length is 767 bytes
```

在 Migrate 過程中出現這個問題的解決方法有兩種

1. 更新到 MySQL 5.7.7 以上版本
2. 調整 Laravel 的預設資料長度

升級 MySQL 的方法就不細說了，接下來主要說明調整 Laravel 預設值的方法

首先，編輯專案目錄下的 `app/Providers/AppServiceProvider.php`

新增一個引用

```
use Illuminate\Support\Facades\Schema;
```

接著修改 boot function，將預設長度定為 191

```
public function boot()
{
    Schema::defaultStringLength(191);
}
```

重新執行 `php artisan migrate` 就不會有問題了

最後附上完整的 AppServiceProvider.php

```
namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Illuminate\Support\Facades\Schema;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Schema::defaultStringLength(191);
    }

    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }
}
```

Laravel 解決在 CentOS 7 下 log 檔無法寫入的問題

在 `config/logging.php`

`'permission' => 0775,`

```
'daily' => [  
    'driver' => 'daily',  
    'path' => storage_path('logs/laravel.log'),  
    'level' => 'debug',  
    'days' => 7,  
    'permission' => 0775,  
],
```

在非 homestead 的主機上執行 laravel 時，有時會出現空白畫面，或是以下的錯誤訊息：

```
The stream or file "/xxxx/yyyy/zzzz/storage/logs/laravel-<日期>.log" could not be opened: failed to open stream: Permission denied
```

這是因為我們使用不同的方式執行 laravel 程式，會讓 log 檔有不同權限設定，和沒有檔案的寫入權限。

內容目錄

- [解法一：將 logS 目錄及檔案權限直接設為 777 \(不推薦\)](#)
- [解法二：設計好適當的權限](#)
 - [步驟一：設定群組權限](#)
 - [步驟二：修改「網頁伺服器」的預設檔案存取權限](#)
 - [步驟三、修改「命令列」的預設檔案存取權限](#)
 - [步驟四、修改「cron」的預設檔案存取權限](#)
 - [解法二總結](#)
- [參考資料](#)

解法一：將 logS 目錄及檔案權限直接設為 777 (不推薦)

```
chmod -R 777 storage/logs/
```

PHP

這方法最快，但是只能治標，不能治本，有再度發生的可能；而且，還有安全性的隱憂

解法二：設計好適當的權限

以 CentOS 為例，在 CentOS 中，我們以 apache 的 virtual host 來執行我們的 laravel 系統，會有三個比較常見的狀況：

- 一、透過「網頁伺服器」執行程式
- 二、透過「命令列」執行程式
- 三、透過「cron」執行程式

各會有不同的檔案權限規則應用在新產生的檔案。我們必須針對這三種方式，來規劃不同的權限規則。

步驟一：設定群組權限

由於 網頁伺服器的預設執行帳號是 apache，所以其產生的檔案，owner及group 均是 apache；而命令列或 cron，則會根據登入的使用者帳號來設定，為了要讓不同方式產生的檔案，能透過「群組權限」的設定值來存取程式，我們必需做一些設定，假設我們 virtual

host 的網頁檔案是以 john 這個帳號來登入的

```
# 把 john 加入 apache 群組
usermod -a -G apache john
# 也把 apache 加入 john 群組
usermod -a -G john apache
```

Bash

更改群組後，shell 要重新登入；部分服務需重新開機才能套用群組權限設定

如果對於 Linux 的檔案權限不太熟悉，可以看一下[鳥哥的文章 - Linux 檔案與目錄管理](#)。

步驟二：修改「網頁伺服器」的預設檔案存取權限

apache server 預設的執行帳號是 apache，產生的檔案權限為 644，範例如下：

```
-rw-r--r-- 1 apache apache 153438 12月 24 20:49 laravel-2019-12-24.log
```

我們必須設定 apache 的 umask 設定，讓其產生的檔案權限為 664，這樣一來，apache 群組才具有寫入的權限。

設定檔 umask.conf 內容

```
[Service]
UMask=0002
```

指令如下：

```
mkdir /etc/systemd/system/httpd.service.d
vi /etc/systemd/system/httpd.service.d/umask.conf

systemctl daemon-reload
systemctl restart httpd.service
```

Bash

步驟三、修改「命令列」的預設檔案存取權限

有時我們會透過 php artisan 來執行程式，產生的檔案，其使用者和群組就是登入的帳號，而權限會根據 umask 來決定其權限。

```
# 在 shell 中，直接輸入 umask 可以查看目前設定
umask

# 修改 umask 為 0002
umask 0002

# 將 umask 0002 的設定加入登入設定檔
vi ~/.bashrc
```

Bash

步驟四、修改「cron」的預設檔案存取權限

當透過 cron 來進行工作排程時，其所產生的檔案，owner 和 group 是登入帳號，預設的 umask 是，0022，我們可以透過修改設定檔來改變：

```
# 修改設定檔，在[Service] 段中，加入 UMask=0002
vi /usr/lib/systemd/system/crond.service

systemctl daemon-reload
systemctl restart crond.service
```

Bash

重新啟動 cron 服務後，其所產生的檔案權限就會變為 664

解法二總結

解法二的處理方式是要達到 2 個目標

1、三種執行方式所產生的 log 檔案，不管 owner 是 apache 或是 john，其權限都是 664

2、apache 和 john 這兩個帳號，都加入對方的群組中

來源：<https://kirin.idv.tw/laravel-log-stream-open-failed-in-centos-7/>

Laravel jwt登入驗證

Laravel API 系列教程（二）：[符合 Laravel 5.5 和 Vue SPA 基于 jwt-auth 实现 API 接口](#) | [构建 API 接口：原生开发](#) | [Laravel 入门到精通教程 \(laravelacademy.org\)](#)

```
#config/app.php
'providers' => [
    #LaravelServiceProvider::class 要改成這個
    Tymon\JWTAuth\Providers\LaravelServiceProvider::class,
],
```

Laravel migration 可使用的欄位類型

可使用的欄位類型

資料庫 Schema 生成器包含表格常用的各種欄位類型，如下所列：

| 程式碼 | 說明 |
|---|--|
| <code>\$table->id();</code> | <code>\$table->bigIncrements('id')</code> 的別名 |
| <code>\$table->foreignId('user_id');</code> | <code>\$table->unsignedBigInteger('user_id')</code> 的別名 |
| <code>\$table->bigIncrements('id');</code> | 遞增 ID（主鍵），相當於「UNSIGNED BIG INTEGER」 |
| <code>\$table->bigInteger('votes');</code> | 相當於 BIGINT |
| <code>\$table->binary('data');</code> | 相當於 BLOB |
| <code>\$table->boolean('confirmed');</code> | 相當於 BOOLEAN |
| <code>\$table->char('name', 100);</code> | 相當於帶有長度的 CHAR |
| <code>\$table->date('created_at');</code> | 相當於 DATE |
| <code>\$table->dateTime('created_at', 0);</code> | 相當於 DATETIME，可以指定位數 |
| <code>\$table->dateTimeTz('created_at', 0);</code> | 相當於 DATETIME (帶時區)，可以指定位數 |
| <code>\$table->decimal('amount', 8, 2);</code> | 相當於 DECIMAL，可以指定總位數和小數位數 |
| <code>\$table->double('amount', 8, 2);</code> | 相當於 DOUBLE，可以指定總位數和小數位數 |
| <code>\$table->enum('level', ['easy', 'hard']);</code> | 相當於 ENUM |
| <code>\$table->float('amount', 8, 2);</code> | 相當於 FLOAT，可以指定總位數和小數位數 |
| <code>\$table->geometry('positions');</code> | 相當於 GEOMETRY |
| <code>\$table->geometryCollection('positions');</code> | 相當於 GEOMETRYCOLLECTION |
| <code>\$table->increments('id');</code> | 遞增 ID（主鍵），相當於 UNSIGNED INTEGER |
| <code>\$table->integer('votes');</code> | 相當於 INTEGER |
| <code>\$table->ipAddress('visitor');</code> | 相當於 IP 地址 |
| <code>\$table->json('options');</code> | 相當於 JSON |
| <code>\$table->jsonb('options');</code> | 相當於 JSONB |
| <code>\$table->lineString('positions');</code> | 相當於 LINESTRING |
| <code>\$table->longText('description');</code> | 相當於 LONGTEXT |
| <code>\$table->macAddress('device');</code> | 相當於 MAC 地址 |
| <code>\$table->mediumIncrements('id');</code> | 遞增 ID（主鍵），相當於 UNSIGNED MEDIUMINT |
| <code>\$table->mediumInteger('votes');</code> | 相當於 MEDIUMINT |
| <code>\$table->mediumText('description');</code> | 相當於 MEDIUMTEXT |
| <code>\$table->morphs('taggable');</code> | 相當於加入遞增 UNSIGNED BIGINT 類型的 taggable_id 與字串類型的 taggable_type |

| 程式碼 | 說明 |
|---|--|
| <code>\$table->uuidMorphs('taggable');</code> | 相當於添加一個 CHAR (36) 類型的 taggable_id 欄位和 VARCHAR (255) UUID 類型的 taggable_type |
| <code>\$table->multiLineString('positions');</code> | 相當於 MULTILINESTRING |
| <code>\$table->multiPoint('positions');</code> | 相當於 MULTIPOINT |
| <code>\$table->multiPolygon('positions');</code> | 相當於 MULTIPOLYGON |
| <code>\$table->nullableMorphs('taggable');</code> | 添加一個可以為空版本的 morphs() 欄位 |
| <code>\$table->nullableUuidMorphs('taggable');</code> | 添加一個可以為空版本的 uuidMorphs() 欄位 |
| <code>\$table->nullableTimestamps(0);</code> | timestamps() 方法的別名 |
| <code>\$table->point('position');</code> | 相當於 POINT |
| <code>\$table->polygon('positions');</code> | 相當於 POLYGON |
| <code>\$table->rememberToken();</code> | 添加一個允許空值的 VARCHAR (100) 類型的 remember_token 欄位 |
| <code>\$table->set('flavors', ['strawberry', 'vanilla']);</code> | 相當於 SET |
| <code>\$table->smallIncrements('id');</code> | 遞增 ID（主鍵），相當於 UNSIGNED SMALLINT |
| <code>\$table->smallInteger('votes');</code> | 相當於 SMALLINT |
| <code>\$table->softDeletes('deleted_at', 0);</code> | 相當於為軟刪除添加一個可為空值的 deleted_at 欄位 |
| <code>\$table->softDeletesTz('deleted_at', 0);</code> | 相當於為軟刪除添加一個可為空值的帶時區的 deleted_at 欄位 |
| <code>\$table->string('name', 100);</code> | 相當於指定長度的 VARCHAR |
| <code>\$table->text('description');</code> | 相當於 TEXT |
| <code>\$table->time('sunrise', 0);</code> | 相當於指定位數的 TIME |
| <code>\$table->timeTz('sunrise', 0);</code> | 相當於指定位數帶時區的 TIME |
| <code>\$table->timestamp('added_on', 0);</code> | 相當於指定位數的時間戳記 |
| <code>\$table->timestampTz('added_on', 0);</code> | 相當於指定位數帶時區的時間戳記 |
| <code>\$table->timestamps(0);</code> | 相當於添加可為空值的时间戳記類型的 created_at 和 updated_at |
| <code>\$table->timestampsTz(0);</code> | 相當於添加指定時區的可為空值的时间戳記類型的 created_at 和 updated_at |
| <code>\$table->tinyIncrements('id');</code> | 相當於自動遞增 UNSIGNED TINYINT |
| <code>\$table->tinyInteger('votes');</code> | 相當於 TINYINT |
| <code>\$table->unsignedBigInteger('votes');</code> | 相當於 UNSIGNED BIGINT |
| <code>\$table->unsignedDecimal('amount', 8, 2);</code> | 相當於 UNSIGNED DECIMAL，可以指定總位數和小數位數 |
| <code>\$table->unsignedInteger('votes');</code> | 相當於 UNSIGNED INTEGER |
| <code>\$table->unsignedMediumInteger('votes');</code> | 相當於 UNSIGNED MEDIUMINT |
| <code>\$table->unsignedSmallInteger('votes');</code> | 相當於 UNSIGNED SMALLINT |
| <code>\$table->unsignedTinyInteger('votes');</code> | 相當於 UNSIGNED TINYINT |
| <code>\$table->uuid('id');</code> | 相當於 UUID |
| <code>\$table->year('birth_year');</code> | 相當於 YEAR |

Laravel queue 使用supervisor 實現多執行序

<https://segmentfault.com/a/1190000021165798>

```
[program:laravel-queue-work]
process_name=%(program_name)s_%(process_num)02d
directory=/data/yoursite
command=php artisan queue:work
autostart=true
autorestart=true
user=www
numprocs=32
redirect_stderr=true
```

安裝supervisor

```
yum install -y supervisor
```

修改設定檔 vim /etc/supervisord.conf

```
[include]
files = supervisord.d/*.conf
;files = supervisord.d/*.ini 改成 files = supervisord.d/*.conf
```

新增 laravel-queue-work.conf

```
vim /etc/supervisord.d/laravel-queue-work.conf
```

```
[program:laravel-queue-work]
process_name=%(program_name)s_%(process_num)02d
directory=/usr/share/nginx/html/laravel5/webmon
command=php artisan queue:work
autostart=true
autorestart=true
user=nginx
numprocs=50
;stdout_logfile = /etc/supervisor.d/log/laravel-queue.log
;stderr_logfile = /etc/supervisor.d/log/laravel-queue_err.log
redirect_stderr=true

[supervisord]
```

常用指令

```
#執行 supervisor
#supervisord
#supervisord -c /etc/supervisord.d/laravel-queue-work.conf (指令config)
[root@aaa]# supervisord
/usr/lib/python2.7/site-packages/supervisor/options.py:461: UserWarning: Supervisord is running as root and it is searching for its
configuration file in default locations (including its current working directory); you probably want to specify a "-c" argument specifying
an absolute path to a configuration file for improved security.
'Supervisord is running as root and it is searching '
```

```
#顯示執行 目前沒有執行
[root@a1-gcm12 etc]# supervisorctl status
unix:///var/run/supervisor/supervisor.sock no such file

#顯示執行 (目前執行)
[root@a1-gcm12 etc]# supervisorctl status
laravel-queue-work:laravel-queue-work_00  RUNNING  pid 28389, uptime 0:00:20
laravel-queue-work:laravel-queue-work_01  RUNNING  pid 28388, uptime 0:00:20
laravel-queue-work:laravel-queue-work_02  RUNNING  pid 28391, uptime 0:00:20
```

#修改設定檔後重新載入

```
[root@a1-gcm12 etc]# supervisorctl reload
```

#停用 supervisor

```
[root@aaa]# supervisorctl shutdown
```

Shut down

Laravel Repository

Day13-[Laravel 資料夾目錄與內容] Repository - iT 邦幫忙::一起幫忙解決難題，拯救 IT 人的一天 (ithome.com.tw)

Laravel_解決CORS錯誤

已封鎖跨來源請求: 同源政策不允許讀取 http://192.168.1.1/api/XXXXXX 的遠端資源。(原因: 缺少 CORS 'Access-Control-Allow-Origin' 檔頭)

建立 app/Http/Middleware/Cors.php

```
<?php

namespace App\Http\Middleware;

use Closure;

class Cors{
    public function handle($request, Closure $next)
    {
        return $next($request)->header('Access-Control-Allow-Origin', '*')
        ->header('Access-Control-Allow-Methods', '*')
        ->header('Access-Control-Allow-Headers', 'Origin, Methods, Content-Type, Authorization')
        ->header('Access-Control-Allow-Credentials', true);
    }
}
```

app/Http/Kenek.php

```
protected $routeMiddleware = [
    //加入
    'cors' => \App\Http\Middleware\Cors::class,
];
protected $middlewarePriority = [
    //加入
    \App\Http\Middleware\Cors::class
];
```

路由加入middle ware

routes/api.php

```
Route::group(['middleware' => 'cors'],function(){
    Route::match(['get','post'],'test/TestApi','TestController@TestApi');
});
```

Laravel_log_permission

config/logging.php

```
'daily' => [  
    'driver' => 'daily',  
    'path' => storage_path('logs/laravel.log'),  
    'level' => 'debug',  
    'days' => 7,  
    'permission' => 0664,  
],
```

参考：[Laravel daily log created with wrong permissions](#) | Newbedev

Laravel_Queue

20分鐘學會Laravel的Queue功能 (pandalab.org)

修改設定檔 env.php -> 指定何種連接

queue.php 連接設定

Slide by : 哥布林工程師

廚師

負責執行訂單

對應:Queue Worker

菜單

構成訂單的最小單位

對應: Job

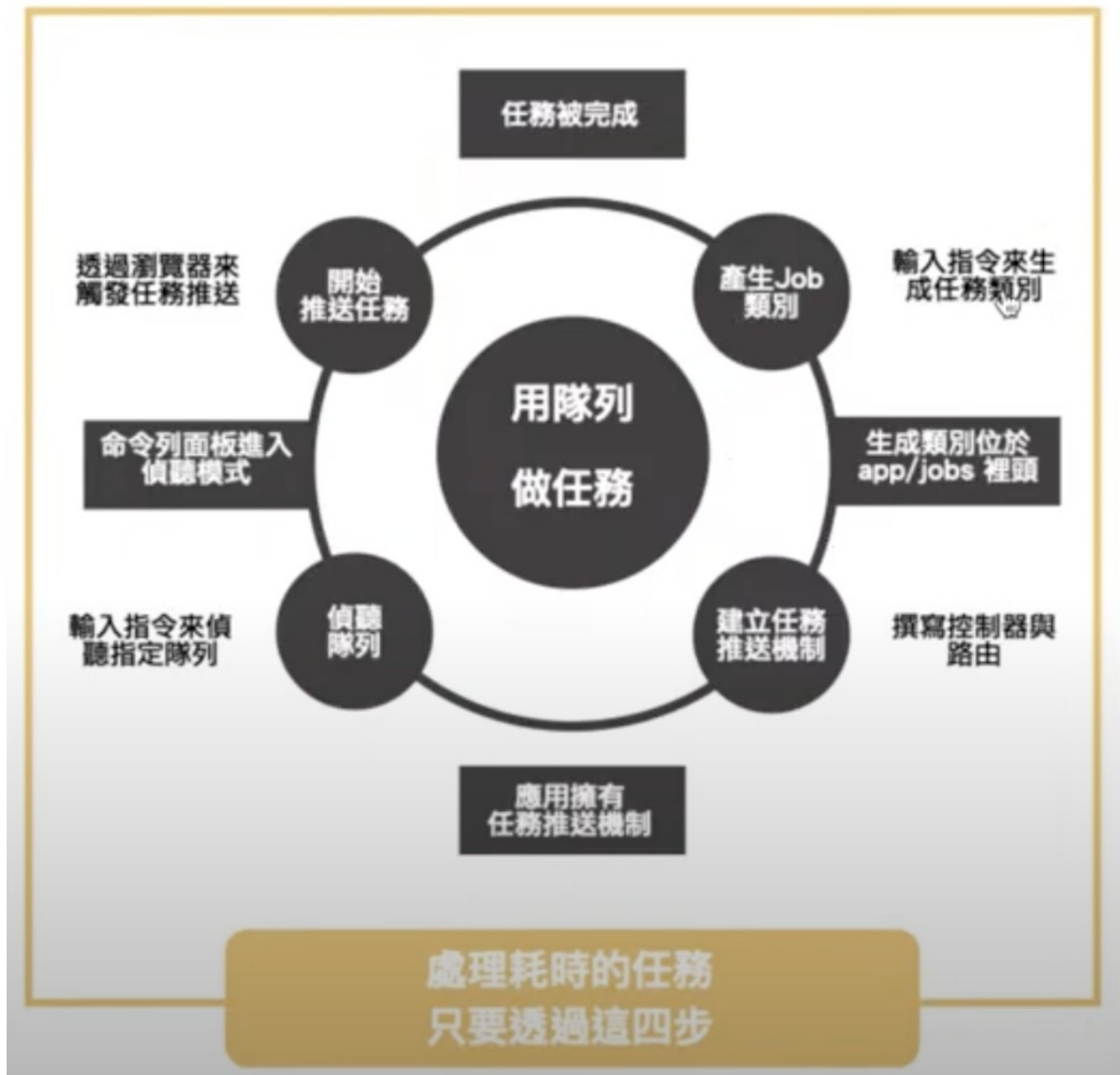
訂單

紀錄要製作哪些菜單

對應: Queue

進廚房

秒懂隊列相關名詞



建立table

```
# 建立 jobs table
php artisan queue:table
# 建立 failed-table
php artisan queue:failed-table

# 執行建立工作
php artisan migrate
```

建立job

```
#php artisan make:job {job name}
php artisan make:job TestGcm
```

Laravel_Request

參考：

[學習LARAVEL 請求，看這篇就夠了 \(pandalab.org\)](https://pandalab.org/learning-laravel-request/)

Laravel Validation

參考:

[Laravel Validation 經驗談. 如何確認 request body的參數是符合我們預期的? ... | by Kidd Chan | K88D | Medium](#)

[Laravel 台灣翻譯文件 | Laravel 道場 \(laravel-dojo.com\)](#)

【Larvel】 redirect() 與 redirect()->intended()

在 Laravel 中，`redirect()->intended('/login')` 和 `redirect('/login')` 有一些差異，主要是用於不同的用途和情境：

1. **`redirect()->intended('/login')`：**

- `intended()` 方法用於重定向到上一個受保護路由的預期 URL。這通常在使用中間件保護的路由中，當用戶未登入而被重定向到登入頁面時，用於指示登入成功後應該返回的路由。
- 當用戶在未登入狀態下訪問一個需要登入才能訪問的路由時，系統會記錄下該受保護路由的 URL。當用戶成功登入後，使用 `redirect()->intended('/login')` 將用戶重定向到該受保護路由的 URL，這樣可以提供更好的使用者體驗。

```
return redirect()->intended('/login');
```

2. **`redirect('/login')`：**

- `redirect('/login')` 簡單地將用戶重定向到指定的 URL，這個 URL 可以是任何您希望將用戶重定向到的地方，例如登入頁面或任何其他頁面。

```
return redirect('/login');
```

差異及適用情境：

- **使用 `redirect()->intended('/login')`：**
 - 主要用於處理登入成功後的重定向，確保用戶可以返回他們原本想要訪問但需要登入才能進入的頁面。
 - 通常與中間件（如 `auth` 中間件）結合使用，當用戶在未登入狀態下訪問受保護路由時會被重定向到登入頁面，成功登入後，會使用 `intended()` 方法將用戶重定向回原本要訪問的受保護路由。
- **使用 `redirect('/login')`：**
 - 簡單地將用戶重定向到指定的 URL，通常用於靜態的重定向，不涉及登入成功後的動態路由記錄和重定向。

總結：

使用 `redirect()->intended('/login')` 和 `redirect('/login')` 主要取決於您的需求。如果您需要確保用戶在成功登入後返回到原本要訪問的受保護路由，應使用 `intended()` 方法。而如果僅僅需要將用戶重定向到靜態的登入頁面或其他固定的 URL，則可以直接使用 `redirect('/login')`。

Laravel 8 + Vue 3 專案建置 (PHP 7.4 支援)

□ 專案名稱：DemoProject

由於 PHP 7.4 無法使用 Vite，因此本指南使用 **Laravel Mix** 來配置 **Laravel 8 + Vue 3** 專案。

□ 步驟 1：安裝 Laravel 8

```
composer create-project laravel/laravel demo-project "8.*"  
cd insure-next
```

啟動 Laravel 伺服器來確認專案正常運行：

```
php artisan serve
```

□ 步驟 2：安裝 Vue 3

在專案目錄下執行以下指令來安裝 Vue 3 和相關套件：

```
npm install vue@3 vue-loader@next
```

□ 步驟 3：修改 Laravel Mix 設定

開啟專案的 `webpack.mix.js`，修改如下：

```
const mix = require('laravel-mix');  
  
mix.js('resources/js/app.js', 'public/js')  
  .vue()  
  .sass('resources/sass/app.scss', 'public/css')  
  .css('resources/css/app.css', 'public/css');
```

□ 步驟 4：建立必要檔案

1. 建立 `resources/js/app.js`：

```
import { createApp } from 'vue';  
import App from './App.vue';  
  
createApp(App).mount('#app');
```

2. 建立 `resources/js/App.vue`：

```
<template>  
  <div>  
    <h1>Welcome to DemoProject!</h1>  
  </div>  
</template>
```

```
</template>
```

```
<script>
export default {
  name: 'App',
};
</script>
```

3. 建立 `resources/sass/app.scss`（如果尚未建立）：

```
mkdir -p resources/sass
```

在 `resources/sass/app.scss` 中新增基本樣式：

```
body {
  font-family: Arial, sans-serif;
  background-color: #f8f9fa;
}
```

4. 建立 `resources/css/app.css`（如果有純 CSS 檔案的需求）：

```
/* app.css */
body {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

▣ 步驟 5：修改 Blade 模板

將 `resources/views/welcome.blade.php` 修改為 `resources/views/index.blade.php`，內容如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>InsureNext</title>
  <link rel="stylesheet" href="{{ mix('css/app.css') }}">
</head>
<body>
  <div id="app"></div>

  <script src="{{ mix('js/app.js') }}"></script>
</body>
</html>
```

▣ 步驟 6：路由設定

在 `routes/web.php` 中新增以下路由設定，將所有頁面導向 Vue 入口頁面：

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

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

❑ 步驟 7：編譯資源

執行以下指令來編譯前端資源：

```
npm run dev
```

若要進行生產環境的編譯，執行：

```
npm run production
```

❑ 錯誤排除 1：Module not found: Error: Can't resolve 'app.scss'

如果你遇到以下錯誤：

```
Module not found: Error: Can't resolve 'resources/sass/app.scss'
```

請確保已建立 `resources/sass/app.scss`，或者修改 `webpack.mix.js`，移除 `sass` 編譯：

```
const mix = require('laravel-mix');

mix.js('resources/js/app.js', 'public/js')
    .vue();
```

❑ 錯誤排除 2：Unable to locate Mix file: /css/app.css

如果你遇到以下錯誤：

```
Unable to locate Mix file: /css/app.css
```

請依照以下步驟處理：

1. 確認資源已正確編譯：

```
npm run dev
```

2. 若問題仍未解決，清除 Laravel 快取：

```
php artisan config:clear
php artisan cache:clear
php artisan view:clear
```

3. 確認檔案 `public/css/app.css` 存在。如果不存在，執行以下指令重新建立檔案：

```
npm run dev
```

❑ 完成！

現在你已經成功建置了一個 **Laravel 8 + Vue 3** 專案，專案名稱為 **InsureNext**。

Laravel 的路由設定已將所有頁面導向 **Vue 入口頁面**，並由 **Vue Router** 來處理前端頁面轉換。