

Laravel Day 1

環境建置 | 建立第一個 Laravel 專案 | 專案結構說明

(內容以 Windows 開發環境為主)

1-1 環境建置

Laravel 是什麼？

- Laravel 是一套用 **PHP** 撰寫的網站框架，幫助我們快速、安全又有邏輯地開發網站與 API。
- 為什麼要學環境建置？因為只有這些工具都準備好，Laravel 專案才能正確運行！

工具	功能是什麼？
PHP	一種程式語言，讓網站會動
MySQL	資料庫，負責存放資料
Composer	幫你的專案安裝外掛和各種套件的小幫手
VS Code	寫程式的軟體（編輯器）
XAMPP	讓網站能在本機跑起來的工具

步驟 1：安裝 XAMPP

- 用途：建立本機開發環境
- 說明：提供 PHP 和 MySQL 等工具，讓我們能在本地測試 Laravel 專案
- 安裝：
 - [官網安裝](#)
 - 安裝位置建議在 C 槽（避免權限錯誤）
 - 安裝流程影片：[How to Download, Install, and Use XAMPP for Windows 10](#)

步驟 2：PHP 中設定環境變數

- 用途：讓你能在任何資料夾直接用 php 指令操作 PHP 程式碼
- 說明：把 PHP 加入系統環境變數後，才可以在任何資料夾下直接輸入 php 指令，不然每次都要切到 PHP 資料夾，會很麻煩
- 步驟：
 - 打開 系統環境變數 設定
 - 在 Path 裡新增 C:\xampp\php
 - 完成後，請重啟終端機
 - 終端機輸入 php -v 確認能正確顯示 PHP 版本資訊
 - 不熟悉的夥伴可參考 [圖文設定教學](#)

步驟 3：安裝 Composer

- 用途：安裝和管理 Laravel 及各種 PHP 套件
- 說明：Composer 就像專案的安裝精靈，幫你下載、管理 Laravel 以及其他功能外掛
- 安裝：
 - [官網安裝](#)
 - 測試安裝是否成功：終端機輸入 `composer --version`，確認能正確顯示 PHP 版本資訊

步驟 4：建立 `checkup.php`

- 說明：確認是否能用 CLI 執行 PHP 程式碼
- 步驟：
 - 創建 `checkup.php`，內容可參考此[範例](#)
 - 在終端機執行：`php checkup.php`
 - 若看到設定畫面，代表成功

步驟 5：修改 php.ini (啟用必要擴充)

- 用途：開啟 Laravel 需要的 PHP 功能
- 說明：有些 PHP 擴充功能（像是 pdo_mysql、fileinfo）預設是關閉的，沒開啟 Laravel 會報錯
- 步驟：
 - 打開：`C:\xampp\php\php.ini`，移除註解（`;`）以啟用：

```
extension=pdo_mysql  
extension=fileinfo
```

- 儲存並重啟 Apache !

1-2 建立第一個Laravel專案

環境安裝設定完成後，我們來建立第一個Laravel專案

建立 Laravel 專案

- 使用 Composer 建立
- 打開終端機輸入以下指令：

```
composer create-project laravel/laravel task-list
```

- 這段指令意思是：用 Composer 幫我新建一個叫 task-list 的 Laravel 專案
- Composer 會自動幫你下載 Laravel 所有需要的東西，建好一個 task-list 資料夾，裡面就是完整的 Laravel 專案。完成後，你只要 cd task-list 進去，就可以開始開發啦！

啟動本地伺服器

```
cd 專案名稱  
php artisan serve
```

✅ Laravel 內建伺服器會啟動，預設網址是：

```
http://127.0.0.1:8000
```

瀏覽器輸入 <http://127.0.0.1:8000> 即可看到 Laravel 首頁 🎉

Artisan 是什麼？

Artisan 是 Laravel 附的指令小幫手，你可以用它做很多事：

- 啟動伺服器（`serve`）
- 建立 controller / model
- 跑 migration / seeder
- 清快取、跑測試
- 指令可參考：[The Laravel Artisan Cheatsheet](#)

試試看 - 動手改首頁

打開這個檔案：

```
routes/web.php
```

把內容改成：

```
Route::get('/', function () {  
    return 'Hello Ruru! 🎉'; #Ruru可以改成你的名字  
});
```

✅ 重新整理瀏覽器 → 看到你打的字就成功囉！

1-3 專案結構說明



Laravel 專案結構概覽

Laravel 專案就像一個網站工地，每個資料夾都是一個負責單位

— app/	← 控制邏輯的大本營
— bootstrap/	← 專案開機程序
— config/	← 系統設定區
— database/	← 資料表、種子資料
— public/	← 網站對外出入口
— resources/	← 畫面 HTML、CSS、JS
— routes/	← 設定網址與功能對應
— storage/	← 上傳資料、log、暫存
— tests/	← 自動化測試
— vendor/	← 套件倉庫 (由 composer 管)

哪些資料夾最常用？

使用頻率	資料夾	用途簡述
✓ 常用	app/	控制器、模型、商業邏輯
✓ 常用	routes/	網址對應功能設定
✓ 常用	resources/	前端畫面、樣式、語系
✓ 常用	database/	資料表結構與假資料
✓ 必看	.env	系統私密設定（不能漏！）
⚙ 較少改	storage/	上傳資料、暫存、快取
⚙ 系統用	vendor/	composer 管理的所有套件

資料夾說明開始！

我們將逐一介紹常見資料夾的用途與開發情境～

app/ 後端主程式

📌 會在這裡建立 Controller、Model，是開發最常碰到的資料夾

- Http/Controllers/：控制器 → 負責接收使用者的操作請求，像是點擊按鈕
- Models/：資料模型 → 對應資料表，幫你存取資料
- Providers/：系統啟動服務載入點

bootstrap/ 啟動設定

📌 通常不需改動，但快取出錯時你可能會來清一清

- app.php : Laravel 核心設定起點
- cache/ : Laravel 自動產生的系統快取資料夾

config/ 系統設定

📌 想改時區、語系、mail 設定 → 就是改這裡！

- 存放 Laravel 所有系統設定檔案
- 每個設定值幾乎都能對應 .env 變數

database/ 資料庫開發專區

📌 建資料表、快速塞資料！

- migrations/：定義欄位／表格結構變化
- factories/：產生假資料的藍圖
- seeders/：把假資料塞進資料表
- database.sqlite：SQLite 資料庫本體

public/ 網站公開入口

 使用者能直接看到的內容都經過這裡

- index.php：整個網站的入口點。
- .htaccess：把所有網址導向 Laravel 的 index.php
- 靜態檔案（圖片 / JS / CSS）可放這裡。

resources/ 畫面 + 多語系

📌 想做一頁「歡迎頁」或「產品列表」，畫面都寫在這裡

- views/ : Blade 模板 (前端畫面)
- lang/ : 語言文字
- css/ / js/ : 前端資源

routes/ 網址與功能對應

📌 想新增 /about 頁面？就是來改 web.php

- web.php：一般網站路由（HTML頁）
- api.php：API 路由（回傳 JSON）
- console.php：CLI 指令定義
- channels.php：事件廣播用設定

storage/ 網址與功能對應

 用 `php artisan storage:link` 把這資料夾公開成網址用

- `app/` : 儲存使用者上傳的檔案。
- `logs/` : Laravel 的錯誤紀錄檔案。
- `framework/` : Session / 快取 等內部資料。

tests/ 自動化測試

📌 要部署上線前跑測試用的區域

- 測試 Controller / API / 資料驗證
- 預設使用 PHPUnit

vendor/ Composer 安裝套件

📌 不需動這裡內容，交給 Composer 管理

- 所有 Laravel 相依的 PHP 套件都裝在這
- 執行 `composer install`、`require` 會更新這裡

根目錄重要檔案

.env

專案私密設定

 這份檔案裡面放著：資料庫帳密、APP 金鑰、寄信資訊... 千萬不能上傳 GitHub！

 實際情境：

- 切換資料庫連線環境（dev / test / production）
- 寄 email、啟用 LINE Notify 等，也都透過這邊設定

`.env.example` 給同事複製的空殼

 是 `.env` 的參考範本，不含帳密，只保留欄位名

 實際情境：

- 新同事 clone 專案後，先複製一份 `.env.example` 改成 `.env`

`.gitignore` Git 忽略清單

🚫 告訴 Git 哪些資料夾 / 檔案 不要上傳

預設會包含：

```
/vendor  
/node_modules  
.env
```

📌 避免私密資訊與巨大套件被意外提交到 GitHub

artisan Laravel 指令列工具

 Artisan 是 Laravel 附的 CLI 工具

```
php artisan serve           # 啟動本地伺服器  
php artisan make:model Product # 建一個 Model  
php artisan migrate         # 執行資料表建立
```

 開發過程會常用，非常重要！

composer.json PHP 套件清單

 定義 Laravel 所使用的所有 PHP 套件，安裝的新套件都會被記錄在這裡！

```
{  
  "require": {  
    "laravel/framework": "^10.0",  
    "barryvdh/laravel-debugbar": "^3.8"  
  }  
}
```

`composer.lock` 實際安裝版本

- 🔒 這是套件版本的「鎖定表單」，確保你跟隊友安裝的一模一樣
- 📌 通常會上傳 Git（跟 `composer.json` 搭配使用）

package.json 前端工具設定

 管理像 Tailwind、Vite、Vue 的 JavaScript 套件

常見指令：

```
npm install  
npm run dev
```

`vite.config.js` Vite 設定檔

⚙️ Laravel 使用 Vite 當作前端打包器，這裡控制打包方式

📌 想加入 Vue 或修改資源位置 → 改這裡

`phpunit.xml` 測試框架設定

 Laravel 使用 PHPUnit 做測試，這是測試的環境與設定檔

 自動化測試會用到，不用碰它，但要知道它存在

README.md 專案說明書

📖 給自己／同事／開源社群看的專案說明手冊，用 Markdown 格式撰寫

內容通常會寫：

- 專案介紹
- 如何安裝
- 常見指令

`.editorconfig` 編碼風格一致

🖌️ 控制專案的排版統一（像是縮排用空白還是 Tab），可跨 IDE、生效於多人協作專案

小結

- 想用 Laravel，這些環境一定要裝好，不然什麼都跑不起來
- 每個資料夾和檔案都有它的用途，知道在哪裡、做什麼事，以後出錯比較不會慌
- 今天只要照步驟做完，你就有自己的 Laravel 專案，可以在本機試著運行看看

接下來的教學內容，就是在這個專案中直接實作囉～