Laravel 專案檔案與資料夾說

解說 Laravel 專案中常見的檔案與資料夾（依照您提供的清單），包括功能、常見內容、開發時要注意的地方與常用指令。適合放在專案 README 的一節或作為新成員的 onboarding 文件。

目錄

根目錄檔案總覽

您提供的根目錄檔案（整理）：

- 資料夾：app, bootstrap, config, database, lang, docker, public, resources, storage, routes, tests, vendor
- 檔案：artisan, composer, composer.json, composer.lock, laravel-echo-server.json, laravel-echo-server.lock, laravel-echo.bat, laravel-file-permission.sh, package.json, phpunit.xml, webpack.mix.js

下面會逐一說明每個資料夾與檔案的用途、常見內容與注意點。

資料夾說明

app

用途：應用程式的主要程式碼放這裡，包含 MVC 的大部分元件（Model、Controller）、Jobs、Events、Console commands、Providers 等。

常見子資料夾： - Console/：自訂 Artisan 指令 - Exceptions/：捕捉及處理例外 - Http/：Controllers/, Middleware/, Requests/（Form Request 驗證） - Models/：Eloquent 模型（有時模型直接放在 app/） - Providers/：ServiceProvider，例如 AppServiceProvider 用來註冊服務、綁定 container - Policies/：授權策略（Authorization）

注意事項： - 遵守 PSR-4 autoload（composer.json 中的 autoload），放檔案到正確 namespace 與資料夾。 - 商業邏輯、資料庫操作應盡量分離（例如用 Services / Repositories pattern）以便測試。

---

bootstrap

用途：應用啟動階段的程式碼與設定。

常見內容： - app.php：建立 Laravel 應用的核心，會回傳 $app 實例（此為 public/index.php 載入的入口）。 - cache/：框架會把一些檔案快取在這裡（例如 routes.php, config.php），以提升啟動效能。

注意事項： - 不要在 bootstrap 放業務程式邏輯。 - 若使用 php artisan config:cache、route:cache，會在這個流程影響 bootstrap/cache 下的快取檔案。

---

config

用途：放框架與套件的設定檔，如 app.php, database.php, mail.php 等。

特色： - 使用 env() 從 .env 讀取機密或環境值 - 可透過 config() helper 讀取設定

注意事項： - 在 production 建議執行 php artisan config:cache，把所有設定快取成單一檔案提升效能（注意：執行後 env() 在執行期間可能無法再動態讀取變更）。 - 不要把機密值硬編在 config 檔內，盡量使用 .env。

---

database

用途：資料庫操作相關檔案。

常見子資料夾： - migrations/：資料庫 schema migration 檔（版本控制資料庫結構） - factories/：Model factory，用於測試或 seeding - seeders/：資料庫填充程式（DatabaseSeeder 可呼叫其他 seeders）

注意事項： - migration 檔是團隊間同步資料庫結構的主要方式，切記妥善命名、版本控制。 - 使用 php artisan migrate --seed 或分開執行 php artisan db:seed。

---

lang

用途：多國語系資源檔（翻譯字串），每個語系一個資料夾，例如 en/, zh-TW/。

內容： - Laravel 的翻譯檔通常為 PHP 檔，回傳陣列（也可使用 JSON 翻譯檔）。

注意事項： - 使用 __('messages.welcome') 或 trans() helper 讀取翻譯內容。

---

docker

用途：放與 Docker 相關的設定與資源（不是 Laravel 官方固定資料夾，但很多專案會建立）。

常見檔案：Dockerfile, docker-compose.yml, 以及環境或 entrypoint script。

注意事項： - 此資料夾結構與內容依專案而異；若使用 Docker 開發或部署，請確認 docker-compose.yml 的 ports、volume 與 DB 設定與 .env 對齊。

---

public

用途：公開可被 HTTP server（Apache/Nginx）直接訪問的目錄。網站根目錄應指向此資料夾。

常見內容： - index.php：所有 HTTP 請求的入口（bootstrap 應用） - favicon.ico, robots.txt - 編譯後放在 public 的靜態資產，例如 js/, css/, images/、有時 mix-manifest.json 或 build/。

重要：不要把敏感檔案放在 public，例如 .env、.git。在部署時 Web server 的 DocumentRoot 應指向 public。

---

resources

用途：前端資源與 view 模板放置地。

常見子資料夾： - views/：Blade 模板（例如 welcome.blade.php） - js/：前端原始碼（Vue、React 或純 JS） - css/ 或 sass/：原始 CSS / Sass 檔 - lang/：有時會放翻譯資源（視專案而定）

注意事項： - npm run dev 或 npm run prod 等資產建置工具會把編譯結果放到 public。 - 若專案使用 Inertia + Vue，Vue component 通常放在 resources/js/Pages 或 resources/js/Components。

---

storage

用途：框架會把可寫入檔案放在這裡，例如使用者上傳檔、快取、session、日誌等（不可直接曝光給網頁訪問）。

常見子資料夾： - app/：應用存檔（有時會放 public/ 子目錄，透過 php artisan storage:link 建立 public/storage 的符號連結） - framework/：session, cache, views 的快取資料 - logs/：Laravel 與應用產生的日誌檔

注意事項： - storage 與 bootstrap/cache 必須是可寫入（web server user 權限）。 - 不要把 storage 放到 public path；若需要公開存取，使用 php artisan storage:link 建 symbolic link 到 public/storage。

---

routes

用途：HTTP 路由設定檔。

常見檔案： - web.php：一般 web 路由（會載入 cookie / session 中介層） - api.php：API 路由（預設 stateless，通常有 api middleware group） - channels.php：廣播頻道授權設定 - console.php：以程式方式定義 Artisan 命令路由

注意事項： - 可使用 php artisan route:list 檢視目前路由清單。 - 若使用 route:cache，動態 closure 路由將無法被快取（使用 controller 的 method 會比較保險）。

---

tests

用途：放 PHPUnit（或 Pest）測試，通常分 Feature/ 與 Unit/。

使用： - 可用 php artisan test 或 vendor/bin/phpunit 執行測試。

注意事項： - 保持測試健全是長期專案品質的關鍵；使用 Factories 建測試資料。

---

vendor

用途：Composer 安裝的所有第三方套件（依賴），由 Composer 自動管理。

注意事項： - 不要把 vendor 加到版本控制（應該在 .gitignore）。 - 若從 repo clone 下來，需執行 composer install 取得 vendor。

重要檔案說明（逐一）

artisan

類型：PHP 可執行的 CLI 檔

用途：Laravel 的命令列工具入口。使用方式例如：

```
php artisan migrate
php artisan serve
php artisan make:controller MyController
```

注意事項： - 在 Windows 上可以直接用 php artisan ...。 - 若 artisan 沒有執行權限，請確認檔案權限或用 php artisan 呼叫。

---

composer（可執行檔）

類型：通常在專案中看到 composer 是指 composer 的可執行檔（或有時是 composer.phar），但較常見的是開發機器全域安裝 Composer，專案內只保留 composer.json 與 composer.lock。

用途：安裝/更新/移除 PHP 套件依賴：

```
composer install
composer update
composer require vendor/package
```

---

composer.json / composer.lock

- composer.json：定義專案的依賴（require / require-dev）、autoload 設定（PSR-4）與 scripts。
- composer.lock：鎖定實際安裝的套件版本，確保團隊或部署環境安裝相同版本。

注意事項： - 在開發時修改 composer.json（例如新增套件）後要 commit composer.json 並執行 composer update 或 composer require，最後也要 commit 產生的 composer.lock。 - 在 CI / 其他開發者機器上，使用 composer install 會依 composer.lock 安裝明確版本。

---

laravel-echo-server.json / laravel-echo-server.lock

用途：當專案使用 Laravel Echo Server（Node.js 的 socket server for broadcasting）時，laravel-echo-server.json 是該伺服器的設定檔，包含 port、authHost、database driver（redis）等設定；.lock 則可能是該伺服器狀態或版本鎖定檔。

注意事項： - 若專案沒有使用 broadcasting 或使用 Pusher 之類第三方，可能看不到這些檔案。 - 啟動 laravel-echo-server 會依該設定檔行為：laravel-echo-server start（或自訂的啟動腳本）。

---

laravel-echo.bat

用途：Windows 平台的 helper / wrapper，協助在 Windows 環境啟動或控制 laravel-echo-server（或其他相關 Node 服務）。

注意事項： - 內容通常是一些批次命令，若在 Windows 開發要調試 socket 功能時會用到。

---

laravel-file-permission.sh

用途：一個 shell script，通常用來在部署時設定 storage、bootstrap/cache 的檔案權限（例如 chown -R www-data:www-data 或 chmod -R 775）。

注意事項： - 這是 Linux / Unix 專用（bash script），在 Windows 上無法直接執行。 - 在部署到 Linux 主機時，執行此 script 可避免權限問題導致應用沒辦法寫入 logs 或 cache。

---

package.json

用途：Node.js / NPM 的依賴與腳本宣告，定義前端套件（如 Vue、Inertia、Vite、Webpack、Laravel Mix）與常用 scripts（例如 dev, prod, watch）。

注意事項： - 當你看到 npm install 後，會產生 node_modules/（通常不會提交到版本控制）。 - 若專案使用 Vite 或 Mix，package.json 裡會有相對應的 build script（例如 vite 或 mix）。

---

phpunit.xml

用途：PHPUnit 的設定檔（測試環境、bootstrap、coverage 設定等）。

注意事項： - CI（如 GitHub Actions）常常會依此檔案執行測試。 - 可定義測試資料庫連線（例如一個 phpunit 專用的 .env.testing），在測試時避免影響生產資料。

---

webpack.mix.js

用途：Laravel Mix（基於 webpack 的抽象）設定檔，用於編譯、打包前端資產（JS、Sass、CSS、images）。

注意事項： - Laravel 後期版本會使用 Vite，但較舊或某些專案仍使用 Laravel Mix（webpack）。 - 若專案有 webpack.mix.js，對應的 package.json scripts 會包含 mix 或 dev、prod 等指令，例如 npm run dev 會執行 mix。

常見命令 & 開發流程小提醒

- 取得相依套件：

```
composer install
npm install
```

- 產生 app key：

```
php artisan key:generate
```

- 建資料庫遷移與 seeder：

```
php artisan migrate
php artisan db:seed
# 或一次完成（視情況）
php artisan migrate --seed
```

- 啟動本地開發伺服器與資產編譯：

```
php artisan serve       # 後端 server（127.0.0.1:8000）
npm run dev             # 前端即時編譯（Vite or Mix）
```

- 常用快取/最佳化：

```
php artisan config:cache
php artisan route:cache
php artisan view:cache
php artisan optimize
```

> 注意：config:cache 與 route:cache 在開發階段可能造成你看不到 .env 或 route 的即時變更，請在部署完成後使用。

- 建立 storage 的公開連結（若需要）

```
php artisan storage:link
```

- 重新產生 autoload（若手動新增 class）：

```
composer dump-autoload
```

安全、部署與注意事項

- **.env 檔案**：不要把 .env 或敏感金鑰放入版本控制（應在 .gitignore）。生產環境應由 CI 或部署系統注入環境變數。
- **vendor/ 與 node_modules/**：通常不提交，改由 composer install 與 npm ci 在部署時產生。
- **資料庫密碼**：盡量使用 secrets 管理（CI / Docker / cloud provider）。

- **檔案權限**：storage 與 bootstrap/cache 必須是可寫（Linux 通常 www-data 或 nginx 使用者）。
- **快取慎用**：在開發環境小心執行 config:cache 與 route:cache，會造成需要清除快取才會看到變更：php artisan config:clear, php artisan route:clear。