

Model 設定與 Seeding

范聖佑 Shengyou Fan
新北市樹林國小 (2015/07/08)

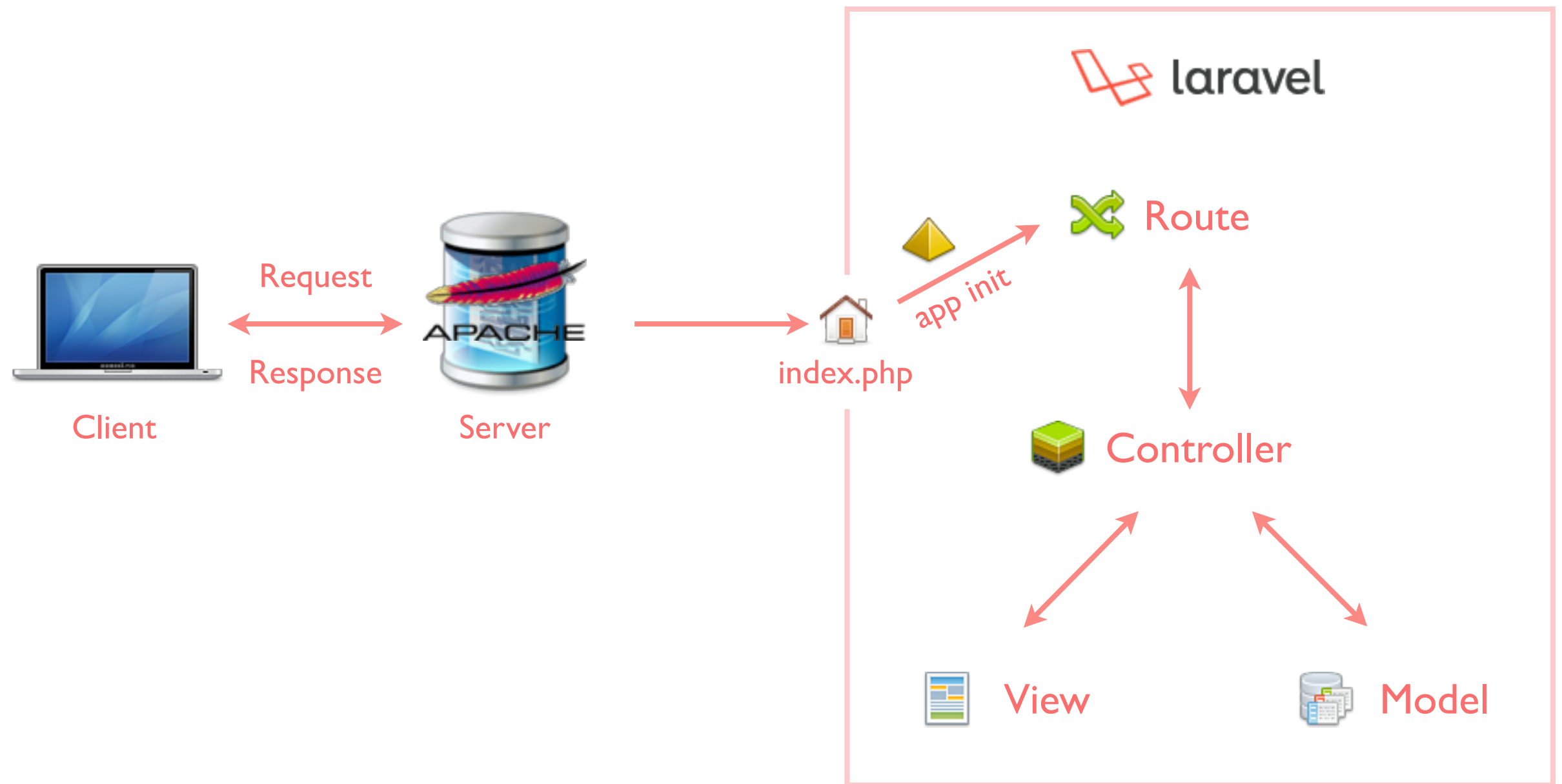
```
1  /**
2   * Display the specified resource.
3   *
4   * @param int $id
5   * @return Response
6   */
7  public function show($id)
8  {
9      $post = Post::with('comments')->where('id', $id)->first();
10
11      return View::make('post.show')->with('post', $post);
12  }
```

單元主題

- 什麼是 Model？什麼是 Seeding？
- 使用 Model 有什麼好處？Laravel 的 Model 機制與實作慣例
- 使用 Seeding 有什麼好處？如何使用 Laravel 的 Seeding 機制與結合 Faker 產生假資料
- Laravel 5.1 的 ModelFactory 功能
- 依照工作坊實作需求示範如何建立 Model 及 Seeder 類別

Model 簡介

Laravel 的 MVC 分工



在有 Model 之前...

- 每當要連線資料庫時，都要自行撰寫連線語法、設定連線編碼、組合 SQL 查詢句、維持/關閉資料庫連結資源 (總之是件苦差事...)

```
// 連線資料庫
$mysqli = new mysqli("DB", "USER", "PASS", "DATABASE");
// 檢查連線
if (mysqli_connect_errno()) {
    /* 略 */
}
// 設定連線編碼
if ($mysqli->set_charset("utf8")) {
    $result = $mysqli->query("/* SQL 查詢句 */");
    $row = $result->fetch_assoc();
    /* 略 */
}
// 關閉連線
$mysqli->close();
```

什麼是 MVC 的 Model ？

- MVC 設計架構裡，M 代表 Model 負責處理資料；V 代表 View 負責顯示頁面；C 代表 Controller 負責控制 M 及 V 間的溝通並回傳結果
- 在此架構設計下，Model 的功能就是負責處理資料，通常就是處理跟資料庫的連線、讀取、寫入等，這些處理資料的邏輯一般會與應用程式的實作目的有關，也就是所謂的業務邏輯
- 如此即可讓 顯示 (V) 與 程式控制 (C)、商業邏輯 (M) 三者獨立開來，彼此分工、互相合作

Laravel 的 Model 機制

- 在 Laravel 的 MVC 分工裡，只要是跟資料讀寫有關的動作，都可以透過 Model 類別來操作。簡單來說，只要是跟資料庫有關的動作，都是使用 Model 來完成
- Laravel 透過實作 ActiveRecord Pattern，讓開發者可以很方便的透過 Model 類別與資料庫溝通，不用每次在跟資料庫溝通時，都要手動撰寫連線語法、SQL 查詢句、處理回傳陣列/物件、維持/關閉資料庫連線狀態等
- 在這個階段先學習如何建立 Model 類別，操作資料的動作細節會在 ORM 一節說明

建立 Model

產生 Model 檔

- 為了讓 Laravel 可以連線至我們所使用的資料庫，務必先設定 `config/database.php` 及 `.env` 裡的資料庫相關連線設定 (上一小節裡已完成)
- 根據 Laravel 的慣例，一個 Model 對應到一個資料表 (table)，所以我們有幾個資料表就會需要幾個 Model 類別檔
- Laravel 的 Model 類別使用大量的慣例，因此一個 Model 類別幾乎只需要繼承父類別、其他什麼都不用設定就可以使用。不過 `artisan` 還是有對應的指令讓我們一行程式碼都不用自己打！

```
$ [php] artisan make:model {name}
```

產生 model 檔案

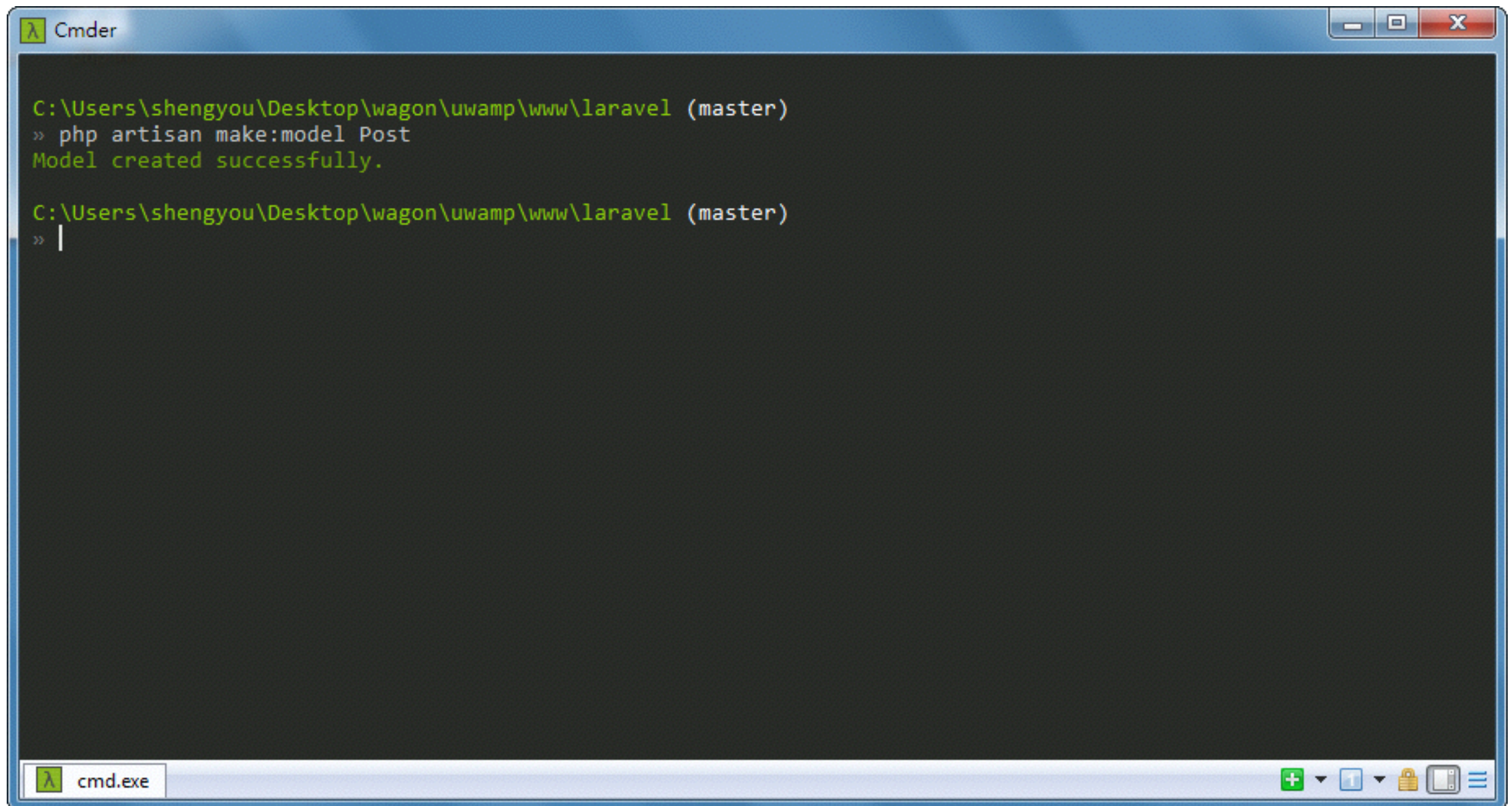
artisan make:model

- 使用 `artisan` 幫我們產生 Model 的類別檔案
 - 自動以傳入的 Model 名稱產生對應的 Model 類別檔案，節省手動輸入程式碼的時間
- 範例：

```
$ php artisan make:model Post
```

執行 make:model

使用 artisan 產生 Model 檔案



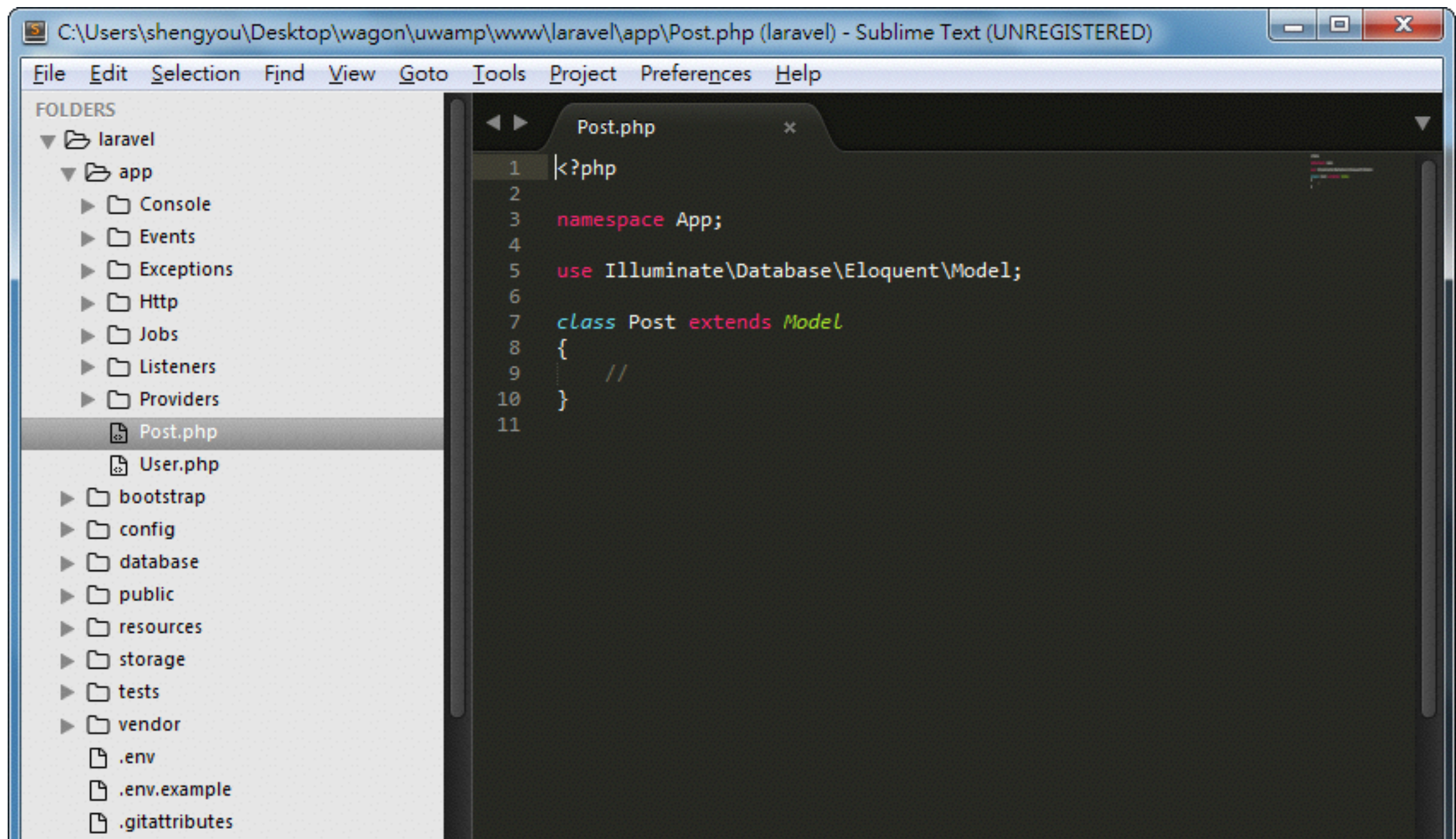
```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan make:model Post
Model created successfully.

C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> |
```

The screenshot shows a Windows Command Prompt window titled 'Cmder'. The window has a blue title bar with standard minimize, maximize, and close buttons. The command prompt shows the current directory as 'C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)'. The user has entered the command 'php artisan make:model Post', and the output is 'Model created successfully.'. The prompt is now waiting for the next command, indicated by a vertical bar cursor.

Model 檔案結構

- 一個 Model 檔只需要非常精簡的幾行結構



Model 慣例

- 產生出來的 Model 檔案，預設會直接放在 `app/` 底下
- 資料表的名稱用英文複數；而 Model 的名稱就用英文單數
- 資料表用英文、全小寫、單字間用蛇底式命名法 (Snake Case)；Model 的名稱用英文、首字大寫、單字間用大駝峰式命名法 (Upper Camel Case)
- 預設的 `namespace` 是在 `App\` 底下，若要更動位置，記得要依 `PSR-4` 更改 `namespace`

指定 Model 對應到的資料表

- Model 與資料表對應的慣例是 Model 的**大駝峰**名稱轉成資料表的蛇底式名稱，若對應不到的話 Laravel 會拋出例外
- 若因特殊原因需要打破這個慣例，可以設定 Model 對應到的資料表名稱

```
<?php
```

```
namespace App;
```

```
use Illuminate\Database\Eloquent\Model;
```

```
class Post extends Model {
```

```
    protected $table = '{table_name}';
```

```
}
```

把 Model 放到資料夾裡

- 隨著專案使用到的 Model 愈來愈多，可能會想要把所有 Model 從 app/ 底下放到 app/Models 底下，記得依以下步驟進行：
 - 先在 app 底下建立 Models 資料夾
 - 把所有 Model 都搬進資料夾內
 - 修正所有 Model 的 namespace 成 **App\Models**

Seeding 簡介

什麼是 Seeding ?

- 當我們在寫資料庫相關的程式時，通常會需要測試資料讀取與顯示是否正常，這時候往往要有一定數量以上的資料才方便開發/測試使用
- **Seed** 就是種子，換成資料庫的概念就是用來做測試的假資料。這些資料只要符合資料庫的欄位屬性即可，但不需要是真的資料
- 現代框架裡所謂的 **Seeding**，就是透過撰寫 **Seeder** 類別來產生假資料，並可用指令將測試資料自動倒入資料庫內。省去手動產生資料或手動倒檔的步驟，就算資料庫清空或是在不同開發環境，也可以很快速的產生測試程式所需要的資料

在有 Seeding 之前...

- 在沒有 Seeding 設計之前，往往需要開發者手動到表單頁自行輸入多次資料，將資料寫入至資料庫
- 更有甚者，若暫時還沒有表單輸入的頁面，就得直接透過資料庫 GUI 將資料一筆一筆的寫入
- 或者是聰明一點，把之前產生出來的假資料 dump 出來成 sql 檔，每次需要時就將 sql 檔重新倒回資料庫內
- 不過若是多人開發時，就會變成除了要交換程式碼之外，也要跟其他開發者索取對應的 sql 檔案

為什麼要用 Seeding ?

- 有測試資料很方便也很重要，但自己產生/建立很花時間，尤其是當這個動作要重複做很多遍的時候是非常累人也非常浪費時間的 (工程師都很懶...)
- 用 Seeding 後，可以快速的讓資料庫內有測試用的假資料，寫資料庫相關的操作功能時，可讓開發效率大增！
- 就算因為測試過程中把資料庫的資料弄亂了，只要再重新產生一次，就可以把資料庫回復到適合測試的狀態

Laravel 的 Seeding 機制

- Laravel 本身就內建了 Seeding 的設計，可以快速的產生測試假資料並倒進資料庫，方便開發以資料操作為主的資料庫相關功能
- Seeder 類別各自獨立，但預設會透過 `DatabaseSeeder` 作為程式執行點，統一呼叫需要執行的 Seeder 一起執行
- `artisan` 提供 `db:seed` 指令，當需要 Seeding 時，可以透過 `artisan` 完成所有資料產生與倒檔的工作

怎麼做 Seeding ?

- 用 `artisan` 產生 Seeder 檔
- 撰寫 Seeder 檔的內容
- 設定 DatabaseSeeder.php 內的執行序
- 使用 `artisan` 執行 `db` 相關指令
- 從 GUI 裡確認執行後的結果
- 確認完成！ (commit 到版本控制系統，需要時就執行)

產生 Seeder 檔

產生 Seeder 檔

- 使用 `artisan` 內建的 `make:seeder` 指令產生 Seeder 檔
- Laravel 沒有特別限制撰寫 Seeder 的規定，不過一般而言會是一個資料表對應一個 Seeder 類別。因此在範例中我們會一個資料表產生一個 Seeder 類別，而後透過 `DatabaseSeeder` 統一呼叫執行


```
$ [php] artisan make:seeder {name}
```

產生 seeder 檔案

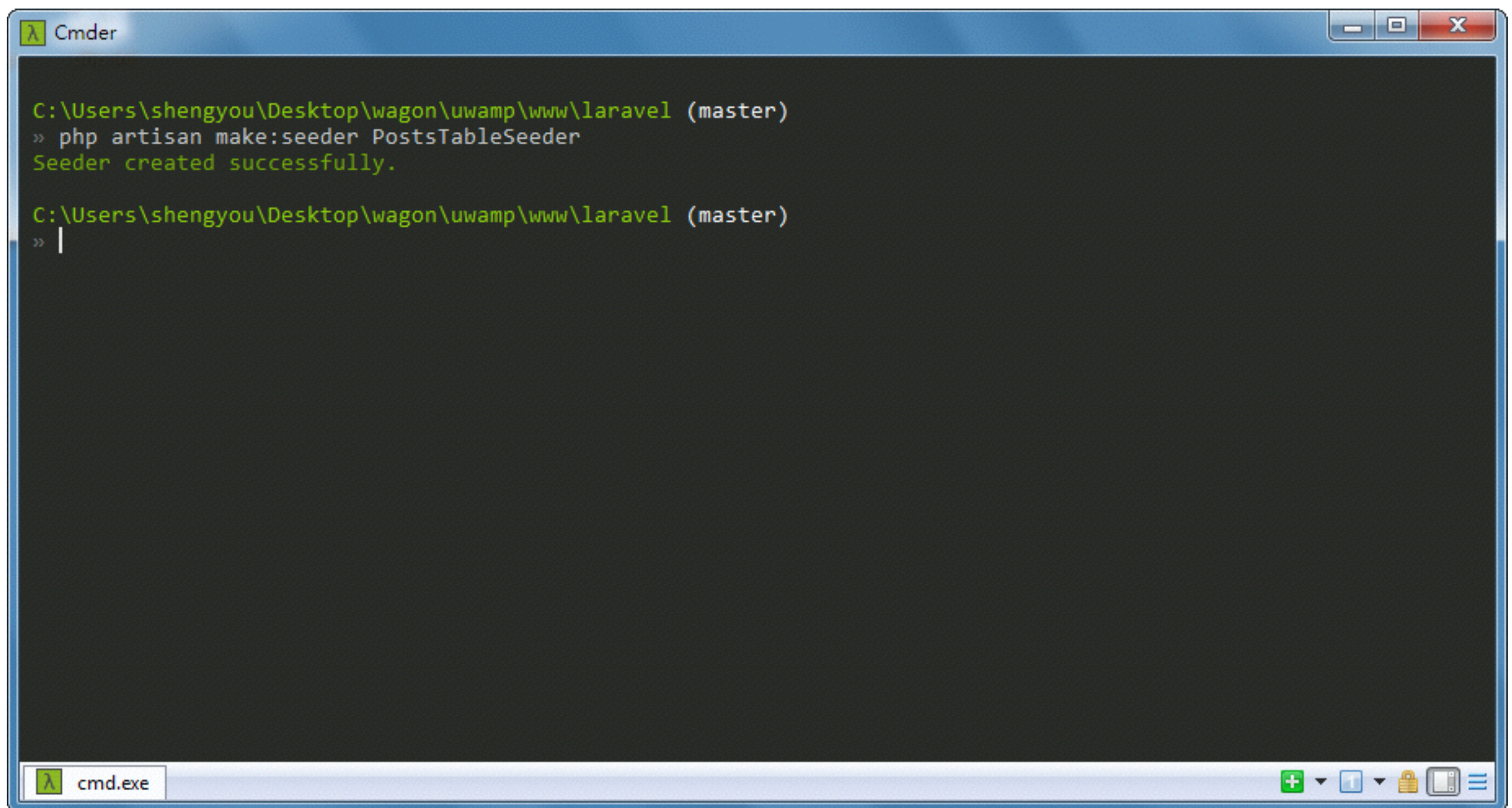
artisan make:seeder

- 透過 `artisan` 產生 Seeder 檔
 - `artisan` 會自動依照給予的名稱，產生對應的 Seeder 類別檔案
- 範例：

```
$ php artisan make:seeder PostsTableSeeder
```

產生 Seeder 檔案

使用 artisan 產生 Seeder 檔案



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan make:seeder PostsTableSeeder
Seeder created successfully.

C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> |
```

The screenshot shows a Windows Command Prompt window titled "Cmder". The window has a blue title bar with standard minimize, maximize, and close buttons. The command prompt shows the current directory as `C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)`. The user enters the command `php artisan make:seeder PostsTableSeeder`, and the output is `Seeder created successfully.`. The prompt then shows the user entering another command, indicated by a vertical bar `|` on the line `>> |`. The taskbar at the bottom shows the "cmd.exe" icon and several system icons on the right.

命名與指令慣例

- 雖然 Laravel 沒有強制規定，但一般命名慣例是使用 {Model 複數名稱}TableSeeder
- 所有的 Seeder 檔案統一放置在 database/seeds/ 資料夾底下

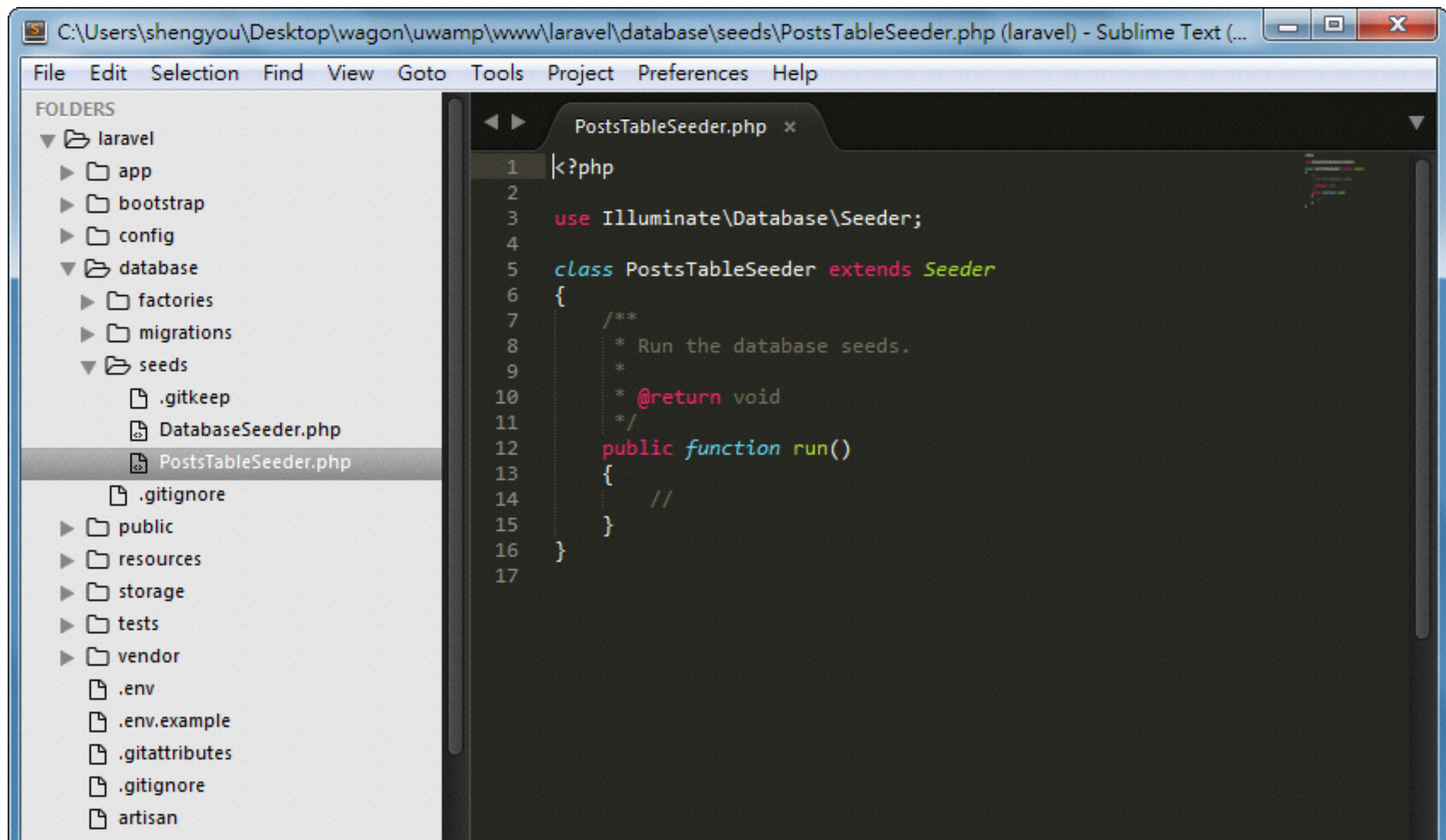
撰寫 Seeder 內容

Seeder 檔結構

- 一個 Seeder 檔裡只有一個 `run` 函式
- 在 `run` 函式裡可以寫任何 Laravel 的指令，不過一般來說會直接操作 DB Facade 和 Model 來新增、寫入、刪除、清空資料庫的資料

Seeder 檔範例

- 用 artisan 產生出來的 Seeder 檔



The screenshot shows a Sublime Text editor window titled "C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel\database\seeds\PostsTableSeeder.php (laravel) - Sublime Text (...)". The left sidebar displays a file explorer with the following structure:

- laravel
 - app
 - bootstrap
 - config
 - database
 - factories
 - migrations
 - seeds
 - .gitkeep
 - DatabaseSeeder.php
 - PostsTableSeeder.php
 - .gitignore
 - public
 - resources
 - storage
 - tests
 - vendor
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - artisan

The main editor area shows the content of "PostsTableSeeder.php":

```
1 <?php
2
3 use Illuminate\Database\Seeder;
4
5 class PostsTableSeeder extends Seeder
6 {
7     /**
8      * Run the database seeds.
9      *
10     * @return void
11     */
12     public function run()
13     {
14         //
15     }
16 }
17
```

撰寫資料產生規則

- 在 run 函式裡直接操作 Model 來產生/寫入資料

```
// database/seeds/PostsTableSeeder.php
class PostsTableSeeder extends Seeder
{
    public function run()
    {
        foreach(range(1, 20) as $number) {
            \App\Post::create([
                'title' => '測試假文章'.$number,
                'sub_title' => '這是副標題...',
                'content' => '這是假的文章內容...',
                'page_view' => 1,
            ]);
        }
    }
}
```


設定 DatabaseSeeder

- 寫好的 Seeder 預設並不會被執行！記得要在 DatabaseSeeder 裡呼叫 (**call**) 才有作用，如此可確保各 Seeder 間的執行順序與相依性

```
class DatabaseSeeder extends Seeder {  
  
    public function run()  
    {  
        Model::unguard();  
  
        $this->call(PostsTableSeeder::class);  
        $this->call(CommentsTableSeeder::class);  
  
        Model::reguard();  
    }  
}
```

unguard、reguard 的功能在 ORM 一節會說明

call 的順序是重要且有相依性的！

執行 Seeding

執行 Seeding

- 撰寫完所需的 Seeder 檔案、並設定好 DatabaseSeeder 後，就可以透過 artisan 的指令來執行 Seeding
- 每一次執行 Seeding，Laravel 就會執行 DatabaseSeeder，並從中依照設定的順序呼叫不同的 Seeder 來產生假資料和寫入資料庫
- 若是 Seeding 過程中有錯誤，記得要自行收拾殘局...

```
$ [php] artisan db:seed
```

執行 Seeding

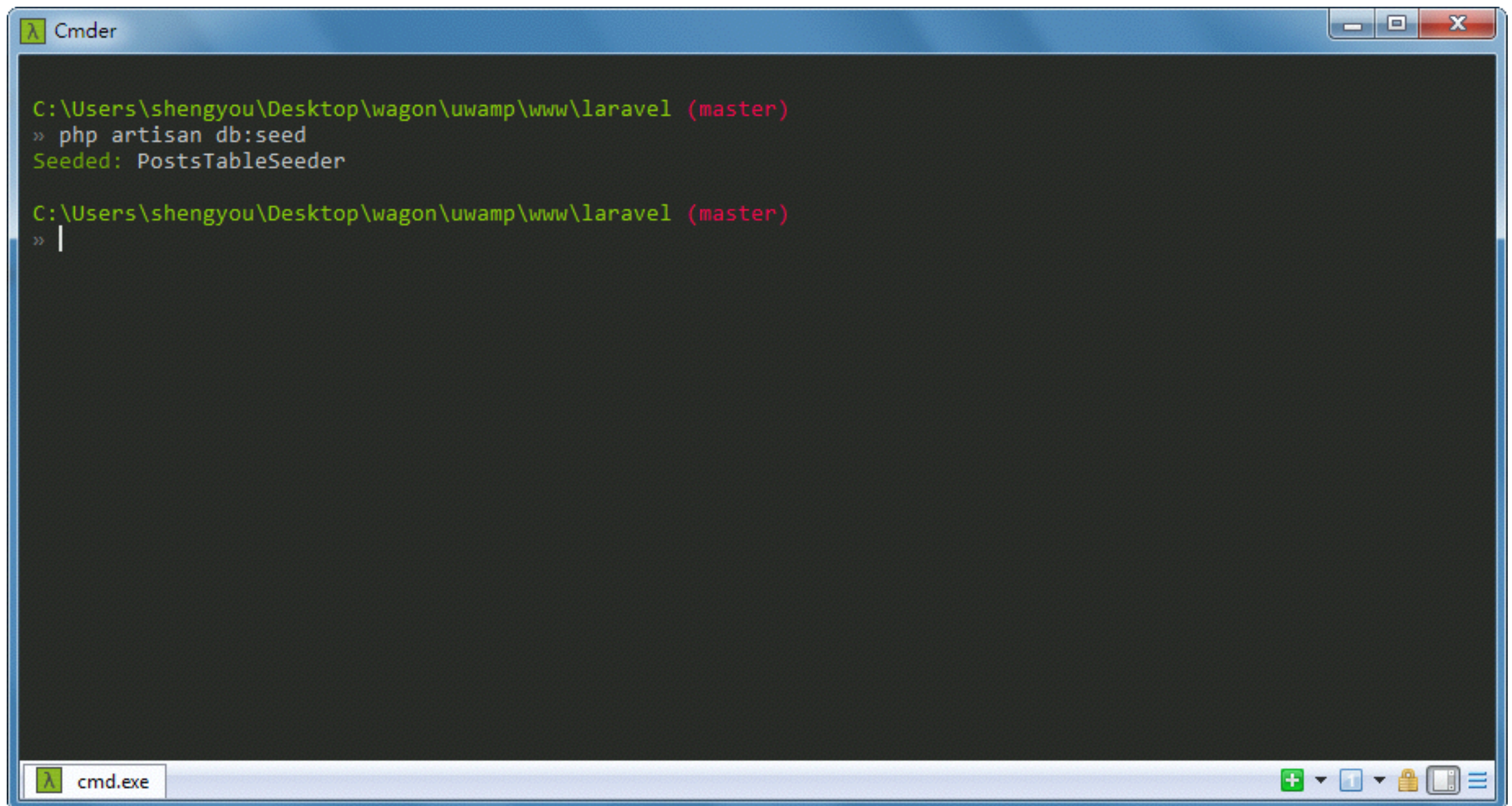
artisan db:seed

- 呼叫 `artisan` 執行 Seeding
 - `artisan` 會執行 `database/seeds/DatabaseSeeder.php` 裡的 `run` 函式，依照函式內部的設定呼叫要執行的 `Seeder` 類別
 - `--class={class_name}` 若只想要執行指定的 Seeding 類別，可加上這個參數來限定
- 範例：

```
$ php artisan db:seed
$ php artisan db:seed --class=PostsTableSeeder
```

執行 Seeding

使用 artisan 執行 db:seed



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan db:seed
Seeded: PostsTableSeeder

C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> |
```

The screenshot shows a Windows Command Prompt window titled 'Cmder'. The current directory is 'C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)'. The user has entered the command 'php artisan db:seed', which has been executed successfully, resulting in the output 'Seeded: PostsTableSeeder'. The prompt is now ready for the next command.

資料庫內驗證

- 用 GUI 工具驗證是否有將資料正確寫入資料庫內

SELECT * FROM 'posts' LIMIT 50 (0.001秒) 編輯

<input type="checkbox"/> Modify	id	title	sub_title	content	page_views	created_at	updated_at
<input type="checkbox"/> 編輯	1	測試假文章1	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	2	測試假文章2	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	3	測試假文章3	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	4	測試假文章4	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	5	測試假文章5	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	6	測試假文章6	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	7	測試假文章7	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	8	測試假文章8	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	9	測試假文章9	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	10	測試假文章10	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	11	測試假文章11	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	12	測試假文章12	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	13	測試假文章13	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	14	測試假文章14	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	15	測試假文章15	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	16	測試假文章16	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	17	測試假文章17	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	18	測試假文章18	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	19	測試假文章19	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48
<input type="checkbox"/> 編輯	20	測試假文章20	這是副標題	這是假文章內容	1	2015-06-21 08:40:48	2015-06-21 08:40:48

(20行) ☐ 所有結果

讓 Seeder 更好用

新增資料前清空資料

- `artisan db:seed` 指令只會單純的執行 `run` 的內容，至於要執行什麼動作、產生什麼結果，完全看程式裡面怎麼寫 (個人造業個人擔！)
- 所以若指令一直下、一直下，假資料就會一直長、一直長，這樣在測試時也會造成其他的困擾
- 我們可以使用 DB Facade 或是 Model 的 `truncate` 指令，在 Seeding 之前先清空資料表，然後再產生假資料寫入
- 範例：

```
DB::table('{資料表名稱}')->truncate();  
\App\Post::truncate();
```

讓資料多點隨機感

- 寫入的假資料都是固定值，測試起來會與真實狀況有所差距
- 在產生資料時，可以利用 `rand` 函式產生隨機數字，讓產生出來的資料可以更隨機一些；也可以使用 `Carbon` 控制日期時間產生間距，讓測試資料不會都被設定在同一天
- 範例：

```
'is_feature' => rand(0, 1);  
'created_at' => \Carbon\Carbon::now()->addDays($number);
```

讓假資料更真！

- 寫入的假資料都很呆板，看起來很不真實。因此可以配合 **Faker** 套件，讓產生出來的測試資料更像真的一樣，會更有 Fu！

```
// database/seeds/PostsTableSeeder.php
public function run()
{
    $faker = \Faker\Factory::create('zh_TW');

    foreach(range(1, 20) as $number) {
        \App\Post::create([
            'title' => $faker->sentence,
            'sub_title' => $faker->sentence,
            'content' => $faker->paragraph,
            /* 略 */
        ]);
    }
}
```

Faker 可以產生的資料類型

- 假文字系
 - \$faker->word
 - \$faker->sentence
 - \$faker->paragraph
- 假身份系
 - \$faker->name
 - \$faker->title
 - \$faker->phoneNumber
 - \$faker->address
- 假網路系
 - \$faker->email
 - \$faker->freeEmail
 - \$faker->userName
 - \$faker->password
 - \$faker->url
 - \$faker->ipv4
 - \$faker->ipv6
 - \$faker->macAddress

Model Factory

Model Factory

- 在 Laravel 5.1 為了更方便開發者整合測試，推出了 Model Factory 功能，讓產生測試資料、撰寫 Seeder 及整合測試流程更加的流暢
- 如何使用 Model Factory ?
 - 在 `databases/factories/ModelFactory.php` 內定義 Model 欄位資料格式
 - 在 Seeder 裡呼叫 `factory()` helper function 並設定相關參與

定義 ModelFactory

- 在 `database/factories/ModelFactory.php` 裡定義每一個 Model 內各欄位的資料產生規則
- 在 `$factory->define()` 裡預設就會把 `Faker` 傳入，所以可以直接使用 `Faker` 的各種 API

```
// database/factories/ModelFactory.php
$factory->define(\App\Post::class, function ($faker) {
    return [
        'title' => $faker->sentence,
        'sub_title' => $faker->sentence,
        'content' => $faker->paragraph,
        'page_view' => rand(0, 20),
    ];
});
```

修改 Seeder 內容

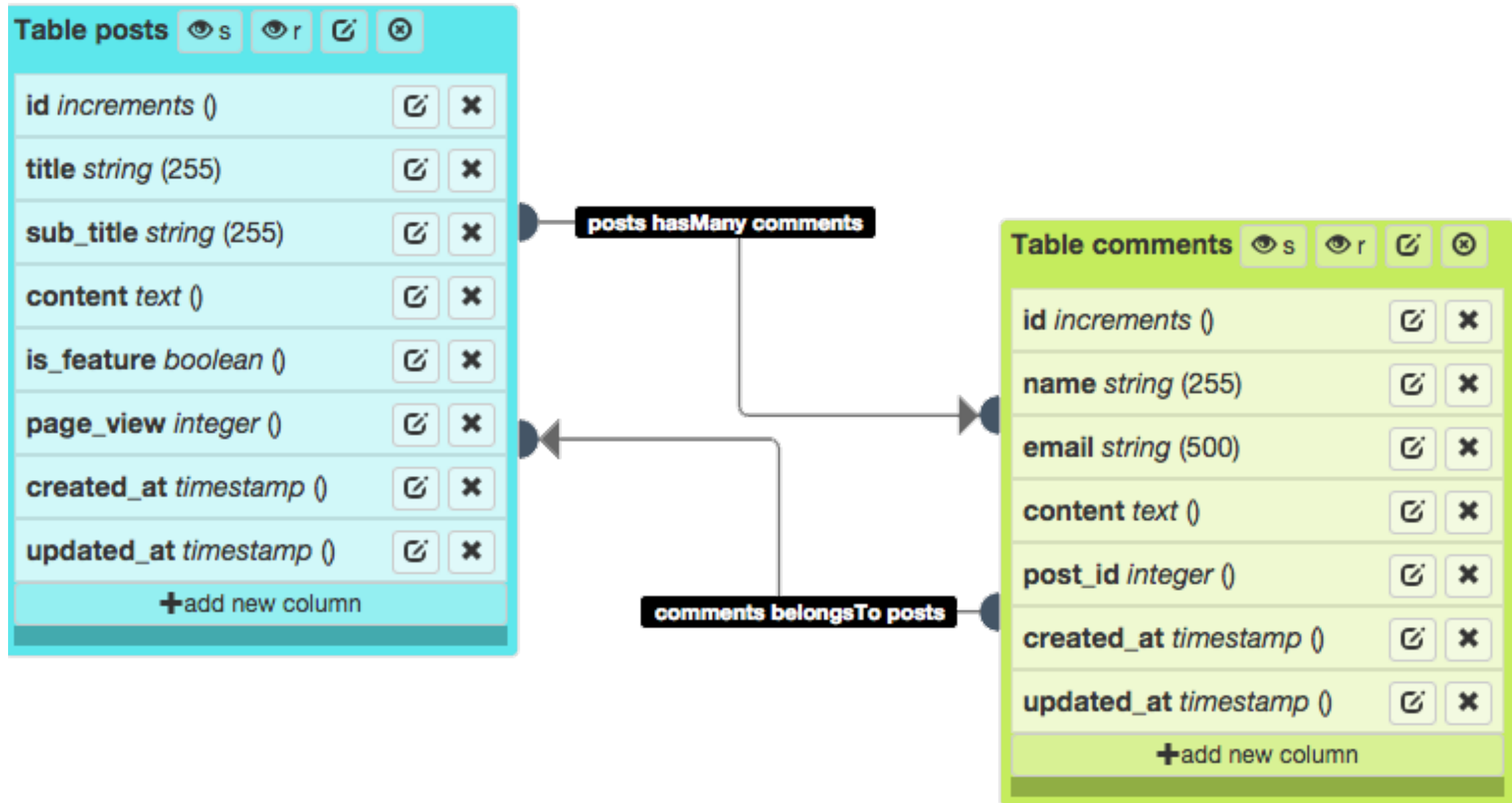
- 原本寫在 Seeder 裡的資料產生邏輯已經被移到 Model Factory 內。因此可以大大簡化 Seeder 檔的內容
- 只需要在 Seeder 內設定清空資料庫及設定要產生的資料筆數即可

```
// database/seeder/PostsTableSeeder.php
public function run()
{
    \App\Post::truncate();

    factory(\App\Post::class, 10)->create();
}
```


實作專案 Seeding 檔

專案資料庫設計



建立 Model/Seeder 檔案

- 針對專案應用程式需要的兩個資料表，用 `make:model` 來產生兩個 Model 檔
- 針對專案應用程式需要的兩個資料表，用 `make:seeder` 來產生兩個 Seeder 檔
- 撰寫 Seeder 檔的內容、設定 ModelFactory
- 設定 DatabaseSeeder 呼叫兩個 Seeder
- 執行 `db:seed`
- 在 GUI 裡檢查結果是否正確

存檔點

- 試著把現在已經可以運作的程式碼加入版本控制內
- 流程提醒：
 - working directory > staging area > commit

單元總結

- 在這個單元裡我們學到了些什麼？
 - Model、Seeding 的基礎概念
 - 如何使用 Model 讓 Laravel 連線資料庫
 - 如何使用 Seeding 及使用 Seeding 的好處
 - 如何在 Seeder 裡使用 Model 寫入資料庫
 - 讓 Seeding 資料更加擬真的技巧
 - 使用 ModelFactory 產生 Seeding 的方式



歡迎提問討論