

Migrations 與 Schema 操作

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 Migration ?
- 使用 Migration 跟以往的作法有什麼不同？
- 對開發者而言，使用 Migration 有什麼好處？
- 如何使用 Laravel 的 Migration ?
- 如何使用 Laravel 的 Schema Builder 操作資料庫？
- 依照工作坊網站規劃書示範如何使用 Migration

Migrations 簡介

什麼是 Migration ？

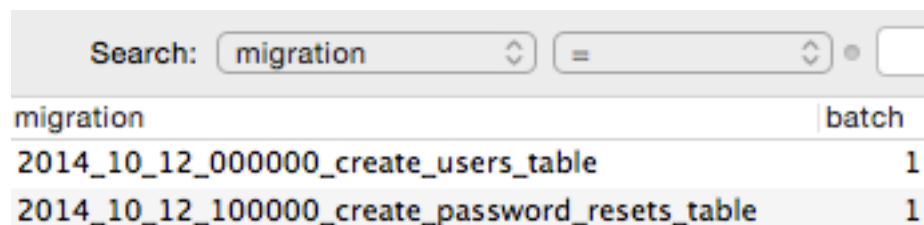
- 相對於手動操作資料庫，Migration 的概念是將所有對 DB 操作的動作，都撰寫成對應的程式碼，透過執行程式來操作資料庫的變更
- 由於動作變成標準的程式碼，因此每個人拿到 Migration 的開發者，就可以透過執行 `migrate` 取得相同的 DB 結構/狀態
- 由於是由程式自動執行，因此可以降低手動操作時可能發生的失誤，若有在變更的過程中發生問題，也可以回覆成先前的狀態
- 簡單來說，可以把它視為資料庫的版本控制系統

在有 Migration 之前...

- 由各開發者手動修改資料庫結構與設定，並把步驟紀錄下來，請其他開發者照著做
- 把修改的步驟 **dump** 成 **sql** 檔案，並把 **sql** 檔傳給其他開發者
- 只要修改的步驟有誤，就得手動檢查發生問題的原因；若在修改之前忘了備份，要回到先前的狀況就很困難
- 若是在自己的開發機上發生錯誤影響較小；但若開發時是共用資料庫，或是在 **staging** 機或 **production** 機上發生時就悲劇了...

Laravel 的 Migration 機制

- `artisan` 提供兩種指令：
 - 產生 migration 檔案的指令
 - 執行 migrate 的指令
- `artisan` 指令除了可以將流程自動化外，也節省開發者撰寫程式碼的時間
- Laravel 會在資料庫內自動產生一個名為 `migrations` 的資料表，在這個資料表內紀錄應用程式 migrate 的歷程 (migration) 及批次 (batch)



The screenshot shows a database table named 'migrations'. It has two columns: 'migration' and 'batch'. There are two rows of data. The first row shows '2014_10_12_000000_create_users_table' in the 'migration' column and '1' in the 'batch' column. The second row shows '2014_10_12_100000_create_password_resets_table' in the 'migration' column and '1' in the 'batch' column.

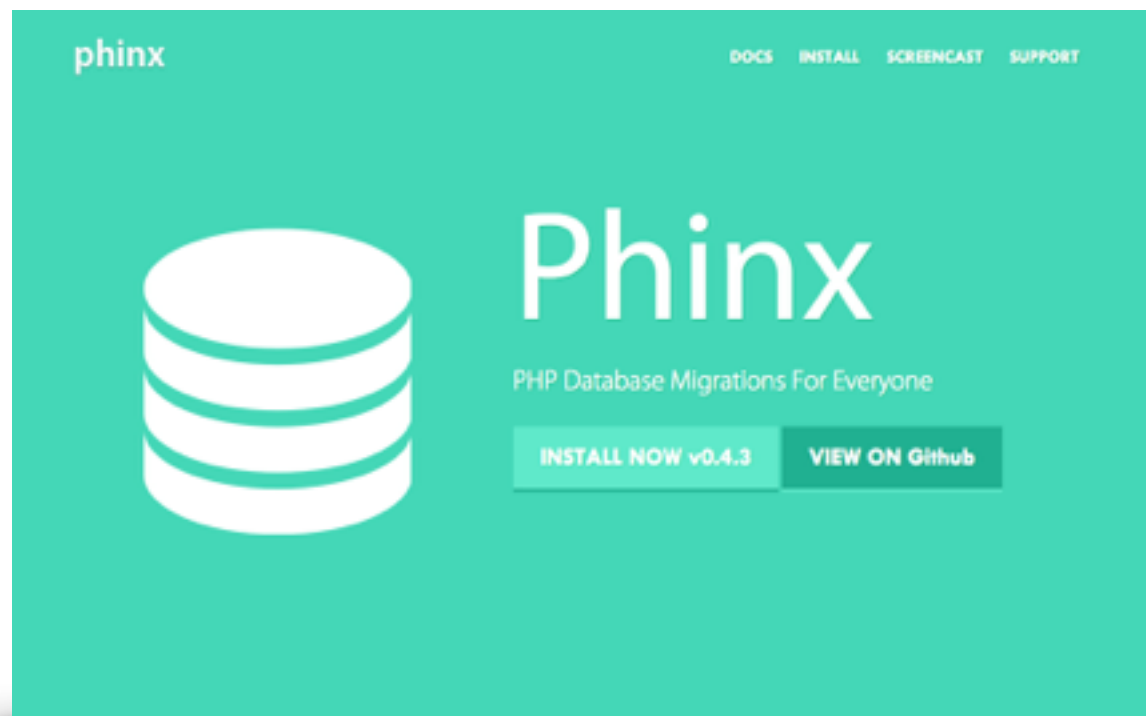
migration	batch
2014_10_12_000000_create_users_table	1
2014_10_12_100000_create_password_resets_table	1

為什麼要用 Migration ?

- 減少錯誤發生的機率
 - 由程式自動化操作資料庫，減少手動操作/人為錯誤發生的可能
- 團隊合作的需求
 - 透過 Migration 達成團隊成員間資料庫同步的需求
- 版本控制的需求
 - 將資料庫的歷次變更紀錄下來，若錯誤發生時，可先回復至先前正確運作的狀態，降低網站 downtime 的時間

若不用 Laravel 的時候...

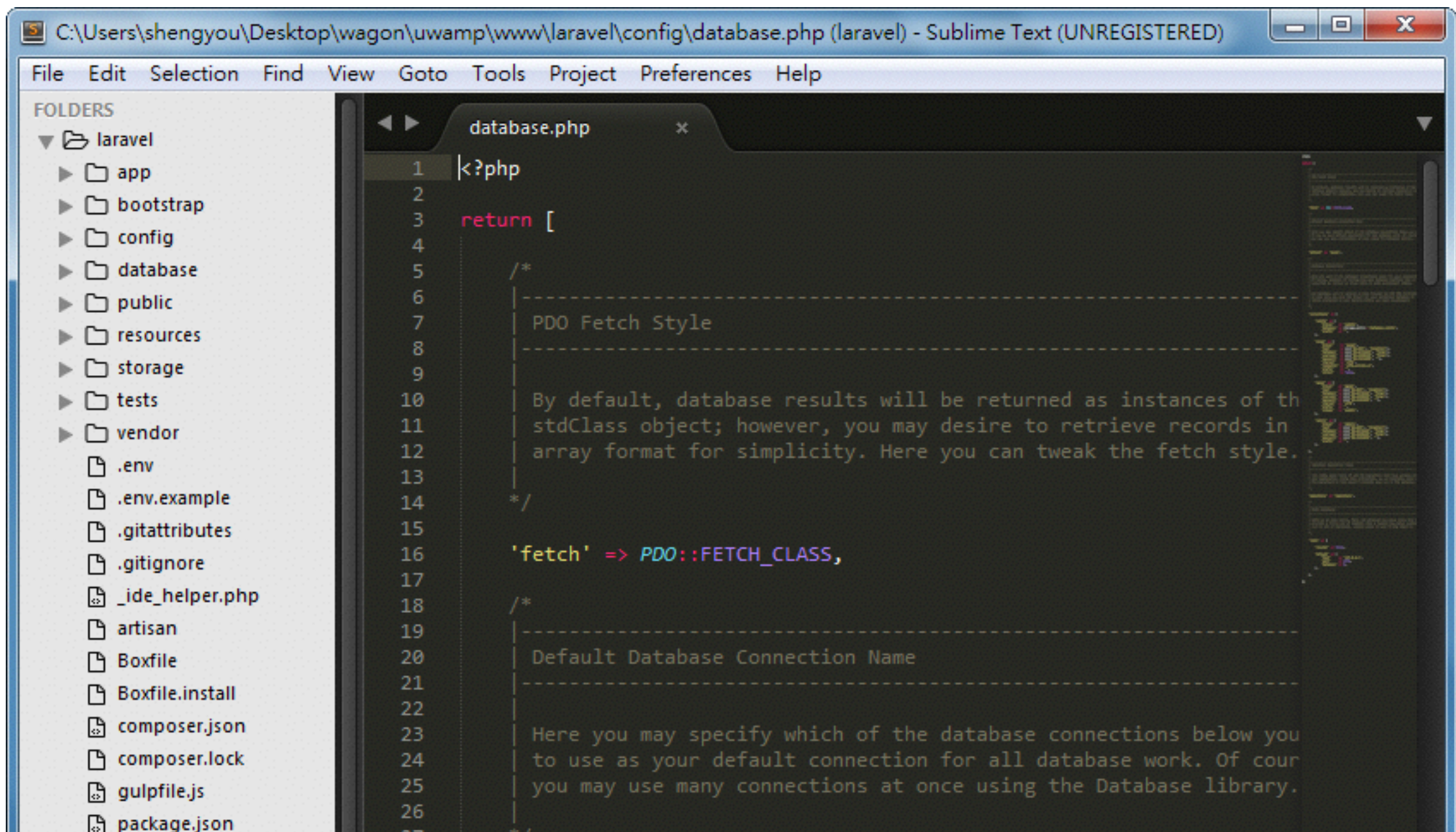
- 即便不是使用 Laravel 開發時，仍可以用其他獨立的 Migration 工具如：
 - [Phinx](#)
 - [dbv.php](#)



Laravel 連線設定

Laravel 的資料庫設定檔

- Laravel 的資料庫設定檔放在 `config/database.php` 裡



設定資料庫連線

```
// config/database.php  
return [
```

```
    'default' => 'mysql',
```

————→ 預設使用的連線

```
    'connections' => [
```

```
        // 略
```

```
        'mysql' => [
```

```
            'driver'
```

```
            => 'mysql',
```

```
            'host'
```

```
            => env('DB_HOST', 'localhost'),
```

```
            'database'
```

```
            => env('DB_DATABASE', 'forge'),
```

```
            'username'
```

```
            => env('DB_USERNAME', 'forge'),
```

```
            'password'
```

```
            => env('DB_PASSWORD', ''),
```

```
        // 略
```

```
    ],
```

```
],
```

```
// 略
```

```
];
```

↓
從 env 檔讀取環境變數

env('{key}', {default})

- Laravel 提供的 helper function 之一
- 在其中 .env 的 **key** 值名稱，就會回傳對應的 **value**
- 若 .env 不存在或 key 值不存在，就會用 **default** 值取代
- 範例：

```
// config/database.php
'database' => env('DB_DATABASE', 'forge'),
'username' => env('DB_USERNAME', 'forge'),
'password' => env('DB_PASSWORD', ''),
```

```
// .env
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret
```

設定資料庫 port

- 假如需要特別設定資料庫的 port，可自行在 `config/database.php` 及 `.env` 裡增加 port 設定

```
// config/database.php
/* 以上略 */
'host'          => env('DB_HOST', 'localhost'),
'port'          => env('DB_PORT', 3306),
'database'      => env('DB_DATABASE', 'forge'),
'username'      => env('DB_USERNAME', 'forge'),
'password'      => env('DB_PASSWORD', ''),
/* 以下略 */
```

```
// .env
DB_HOST=localhost
DB_DATABASE=homestead
DB_PORT=8889
DB_USERNAME=homestead
DB_PASSWORD=secret
```

建立資料庫

- 可用 CLI 或 GUI 連線至資料庫，並建立要給 Laravel 使用的資料庫
- 資料庫命名慣例：
 - 專案名稱_環境，如 blog_local

選擇資料庫

建立新資料庫

權限 處理程序列表 變數 狀態

MySQL版本：5.6.20-log 透過PHP擴充模組 MySQLi

登錄為：root@localhost

	資料庫 - 重新載入	校對	資料表	Size - Com
<input type="checkbox"/>	information_schema	utf8_general_ci	?	
<input type="checkbox"/>	mysql	latin1_swedish_ci	?	
<input type="checkbox"/>	performance_schema	utf8_general_ci	?	

建立資料庫

(校對) 儲存 +

資料庫: laravel_local

已建立資料庫。 09:49:00 SQL命令

修改資料庫 資料庫架構 權限

測試資料庫連線

- 設定完資料庫連線設定後，可以先用 `artisan` 產生 `migration` 要用的資料表
- 透過產生 `migrations` 資料表，也可以做為測試資料庫連線正確的方式

```
$ [php] artisan migrate:install
```

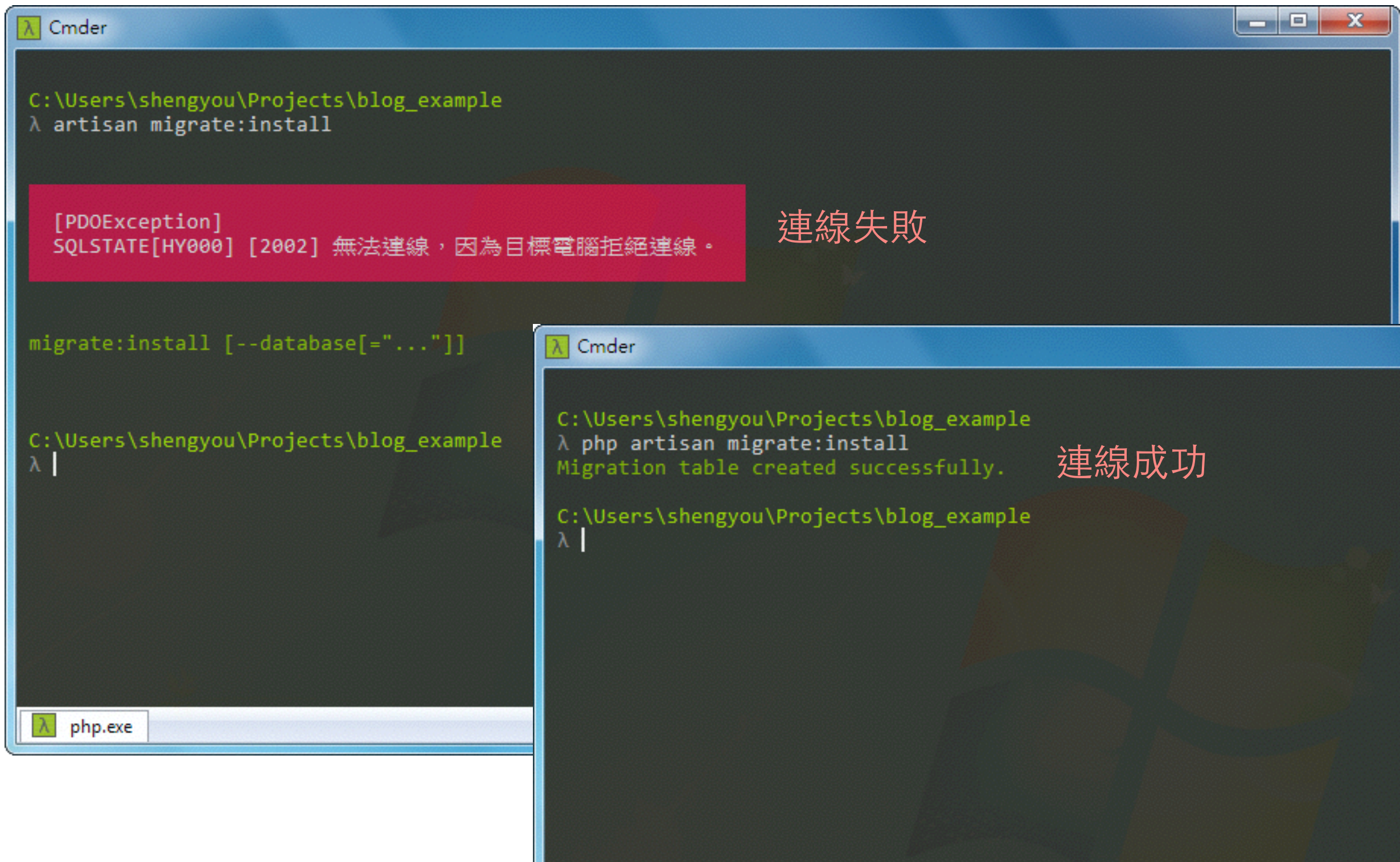
在資料庫內產生 migrations 資料表

artisan migrate:install

- 產生 migrations 資料表
 - artisan 會使用目前的環境設定連線至資料庫，並產生一個名為 migrations 的資料表
- 範例：

```
$ php artisan migrate:install
```

執行 migrate:install



從 GUI 確認結果

The screenshot shows the MySQL Adminer 4.2.1 interface. The browser address bar displays the URL: `localhost:8000/adminer/?server=localhost%3A3306&username=root&db=laravel_local`. The page title is "MySQL » localhost:3306 » 資料庫: laravel_local". The language is set to "繁體中文". The database selected is "laravel_local". The left sidebar shows the "migrations" table selected. The main content area shows the "資料表和檢視表" (Tables and Views) section. A search bar indicates "在資料庫搜尋 (1)". A table lists the tables in the database:

<input type="checkbox"/>	資料表	引擎?	校對?	資料長度?	索引長度?	資料空間?	自動遞增?	行數?	註解?
<input type="checkbox"/>	migrations	MyISAM	utf8_unicode_ci	0	1,024	0		0	
總共 1 個		MyISAM	utf8_general_ci	0	1,024	0			

Below the table, there is a "Selected (0)" section with buttons for "分析", "最佳化", "檢查", "修復", "清空", and "刪除". At the bottom, there is a "轉移到其它資料庫:" dropdown menu set to "laravel_local" and buttons for "轉移" and "複製".

確認產生 migrations 資料表

Migration 實作流程

怎麼做 Migrate ?

- 先用 `artisan` 產生 migration 檔
- 撰寫 migration 檔的內容 (使用 Schema Builder 類別)
- 使用 `artisan` 執行 `migrate` 相關指令
- 從 GUI 裡確認執行後的結果
- 測試 `rollback` 是否正確
- 確認完成！ (commit 到版本控制系統)

產生 Migration 檔

make 系列指令

- artisan 有一系列 make 開頭的指令，可以協助開發者自動化的產生撰寫程式時所需要的 class 檔案

```
$ php artisan
Laravel Framework version 5.1.2 (LTS)
Available commands:
/* 略 */
make
/* 略 */
make:controller  Create a new resource controller class
make:migration   Create a new migration file
make:model       Create a new Eloquent model class
make:request     Create a new form request class
make:seeder      Create a new seeder class
/* 略 */
```

```
$ [php] artisan make:migration \
    {action}_{table}_table
```

產生 migration 檔案

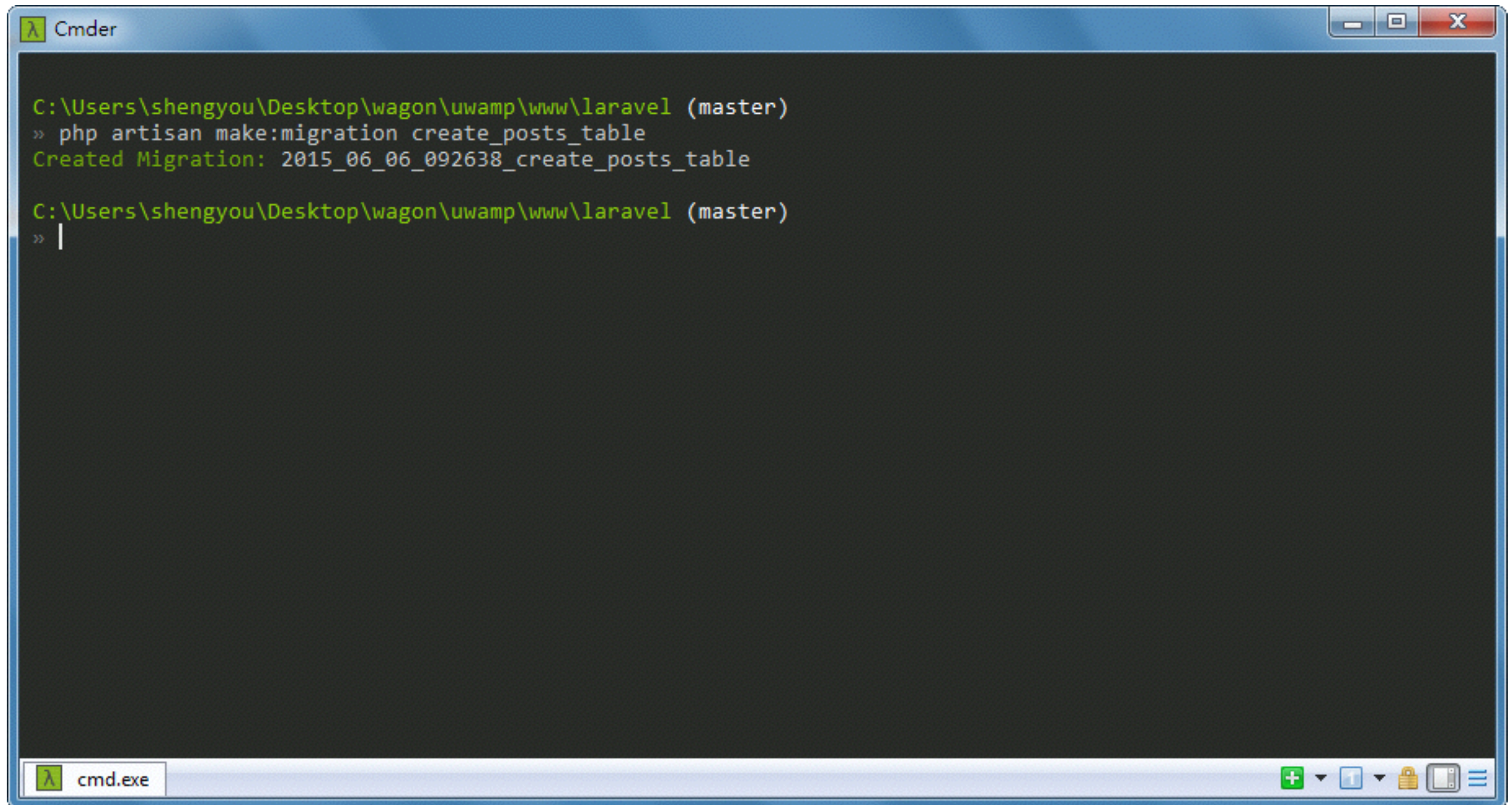
artisan make:migration

- 產生 migration 檔
 - 依照給予的 migration 名稱，產生 migration 檔，並執行 dump-autoload
 - --create={table_name} 要新增的資料表名稱
 - --table={table_name} 要變更的資料表名稱
- 範例：

```
$ php artisan make:migration create_posts_table --create=posts
$ php artisan make:migration add_email_in_users_table \
  --table=users
```

產生 migration 檔

使用 artisan make:migration 產生 migration 檔



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan make:migration create_posts_table
Created Migration: 2015_06_06_092638_create_posts_table

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

命名與指令慣例

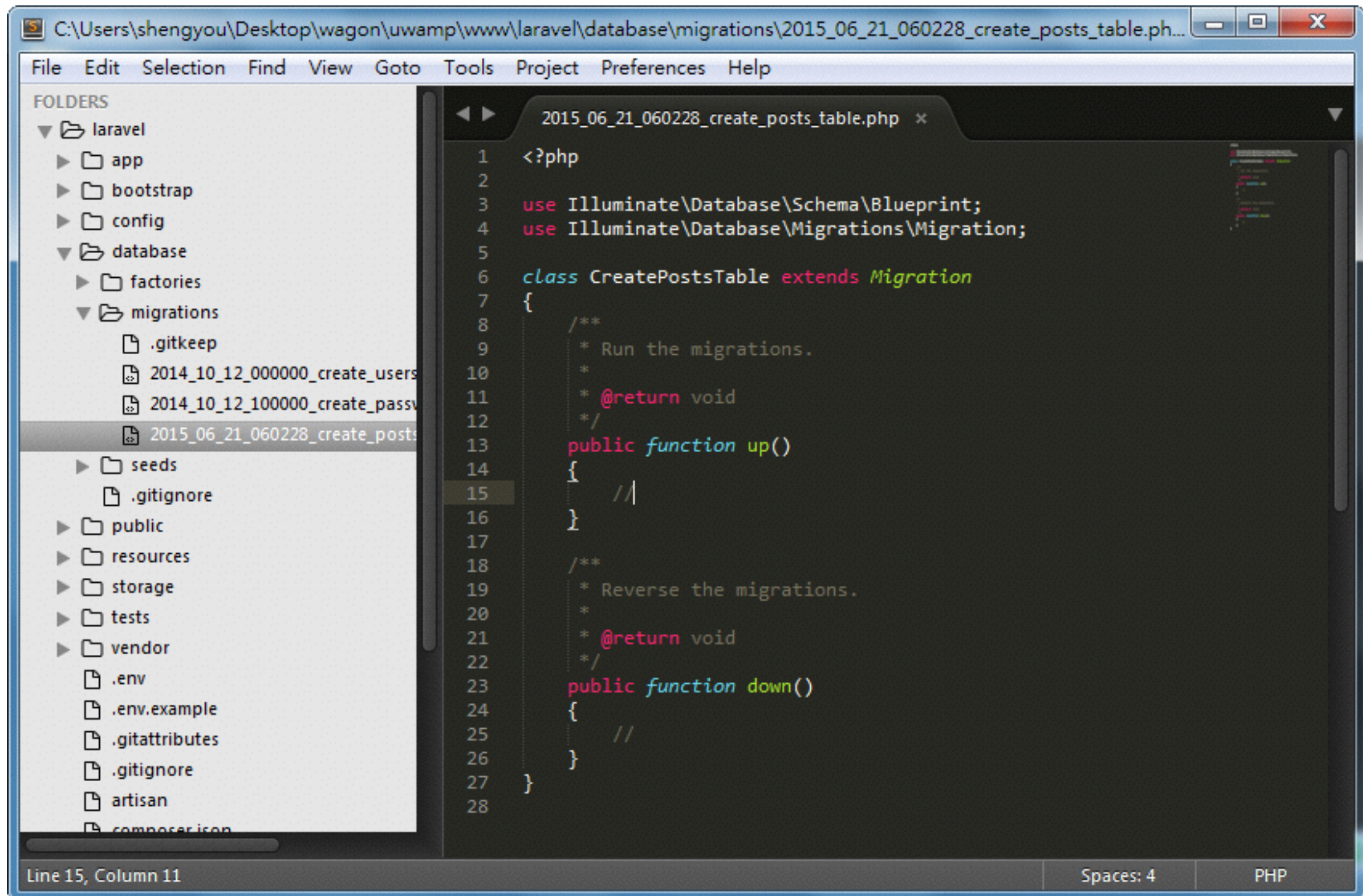
- 為了確保開發者間產生出來的 migration 檔不會有重覆的檔名，artisan 在產生 migration 檔時，會自動把時間戳記放在檔名前 (2016_07_08_092638)
- 通常檔名命名以 動作 + 資料表名稱 做組合，中間以「底線 _」區隔
 - 常見動作 create, add, remove
 - 資料表名稱以複數命名
- 若是新建資料表則加 --create= 參數、若是修改/更新資料表則加 --table= 讓 artisan 幫你多寫一點程式碼

撰寫 Migration 檔

Migration 檔結構

- 一個 Migration 檔裡一定實作兩個 method：
 - **up** (前進)：在這個 method 裡，用 Schema Builder 撰寫要對資料庫做的動作，如新增、刪除、重新命名資料表；修改資料表的欄位名稱、資料類型...等
 - **down** (後退)：相對於 up，down 裡要寫的動作就是 up 的相反，也就是說要寫如何將 up 的內容還原回去，做為版本回溯機制

Migration 檔範例



The screenshot shows a code editor window with a file explorer on the left and a code editor on the right. The file explorer shows the directory structure of a Laravel project, with the 'migrations' folder expanded. The code editor displays the content of the file '2015_06_21_060228_create_posts_table.php'. The code is a PHP migration class that extends 'Migration' and implements 'up()' and 'down()' methods. The 'up()' method is currently empty, indicated by a comment '//'. The 'down()' method is also empty, indicated by a comment '//'. The status bar at the bottom indicates 'Line 15, Column 11', 'Spaces: 4', and 'PHP'.

```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel\database\Migrations\2015_06_21_060228_create_posts_table.ph...
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
└─ laravel
   └─ app
   └─ bootstrap
   └─ config
   └─ database
      └─ factories
      └─ migrations
         .gitkeep
         2014_10_12_000000_create_users
         2014_10_12_100000_create_passw
         2015_06_21_060228_create_posts
      └─ seeds
         .gitignore
      public
      resources
      storage
      tests
      vendor
      .env
      .env.example
      .gitattributes
      .gitignore
      artisan
      composer icon

2015_06_21_060228_create_posts_table.php
1  <?php
2
3  use Illuminate\Database\Schema\Blueprint;
4  use Illuminate\Database\Migrations\Migration;
5
6  class CreatePostsTable extends Migration
7  {
8      /**
9       * Run the migrations.
10      *
11      * @return void
12      */
13     public function up()
14     {
15         //
16     }
17
18     /**
19      * Reverse the migrations.
20      *
21      * @return void
22      */
23     public function down()
24     {
25         //
26     }
27 }
28

Line 15, Column 11 Spaces: 4 PHP
```

Schema Builder

- Laravel 提供 Schema 類別，讓開發者可以抽象用同一種語法操作所有支援的資料庫
- 透過 Schema Facade 可以讓我們：
 - 設定資料表使用的引擎
 - 建立/更名/刪除資料表
 - 新增、更名、刪除資料表欄位
 - 更改資料表欄位屬性、增加/刪除資料表欄位索引
 - 快速設定 ORM 要使用的時間戳記和軟刪除欄位

建立/更名/刪除資料表

- 建立資料表

```
Schema::create('posts', function(Blueprint $table)
{
    //
});
```

- 重新命名資料表

```
Schema::rename($from, $to);
```

- 刪除資料表

```
Schema::drop('posts');
Schema::dropIfExists('posts');
```


新增/更名/刪除欄位

- 建立欄位

```
Schema::table('posts', function(Blueprint $table)
{
    $table->string('email');
});
```

- 重新命名欄位 (要額外安裝 doctrine/dbal 套件)

```
Schema::table('posts', function(Blueprint $table)
{
    $table->renameColumn('from', 'to');
});
```

- 刪除欄位

```
Schema::table('posts', function(Blueprint $table)
{
    $table->dropColumn('abstract');
});
```

資料表欄位設定

- `$table->increments('id');`
- `$table->boolean('confirmed');`
- `$table->integer('votes');`
- `$table->string('name', 100);`
- `$table->text('description');`
- `$table->date('created_at');`
- `$table->dateTime('created_at');`
- `$table->timestamp('added_on');`
- `->nullable()`
- `->default($value)`
- `->unsigned()`

設定時間戳記和軟刪除

- 設定時間戳記

```
Schema::table('posts', function(Blueprint $table)
{
    $table->timestamps(); // 建立時間戳記
    $table->dropTimestamps(); // 刪除時間戳記
});
```

- 設定軟刪除

```
Schema::table('posts', function(Blueprint $table)
{
    $table->softDeletes(); // 建立軟刪除戳記
    $table->dropSoftDeletes(); // 刪除軟刪除戳記
});
```

更改欄位屬性、索引

- 更新欄位屬性

```
Schema::table('users', function(Blueprint $table)
{
    $table->string('email', 500)->change();
});
```

- 增加 index

```
$table->integer('post_id')->index();
$table->string('email')->unique();
```

- after 語法

```
$table->string('email')->after('username');
```

設定資料表使用的引擎

- 設定資料表使用引擎

```
Schema::create('posts', function(Blueprint $table)
{
    $table->engine = 'InnoDB';
});
```

執行 Migrate

migrate 系列指令

- artisan 有一系列 migrate 開頭的指令，透過這些指令來操作資料庫的版本變更

```
$ php artisan
Laravel Framework version 5.1.2 (LTS)
Available commands:
```

```
/* 略 */
```

```
  migrate
```

Run the database migrations

```
/* 略 */
```

```
migrate
```

```
  migrate:install
```

Create the migration repository

```
  migrate:refresh
```

Reset and re-run all migrations

```
  migrate:reset
```

Rollback all database migrations

```
  migrate:rollback
```

Rollback the last database migration

```
  migrate:status
```

Show the status of each migration

```
/* 略 */
```

了解當前 migrate 狀態

- 使用 `migrate:status` 指令，由 `artisan` 回傳當前 `migrate` 狀態
- `artisan` 會將所有的 `migration` 檔以表格的方式印出，並標記各個 `migration` 被執行的狀態
- 從資料庫裡的 `migrations` 資料表也可以觀察出目前 `migration` 被執行的狀態

`SELECT * FROM `migrations` LIMIT 50 (0.001秒) 編輯`

<input type="checkbox"/> Modify	migration	batch
<input type="checkbox"/> 編輯	2014_10_12_000000_create_users_table	1
<input type="checkbox"/> 編輯	2014_10_12_100000_create_password_resets_table	1

(2行) ☐ 所有結果


```
$ [php] artisan migrate:status
```

顯示當下 migrate 狀態

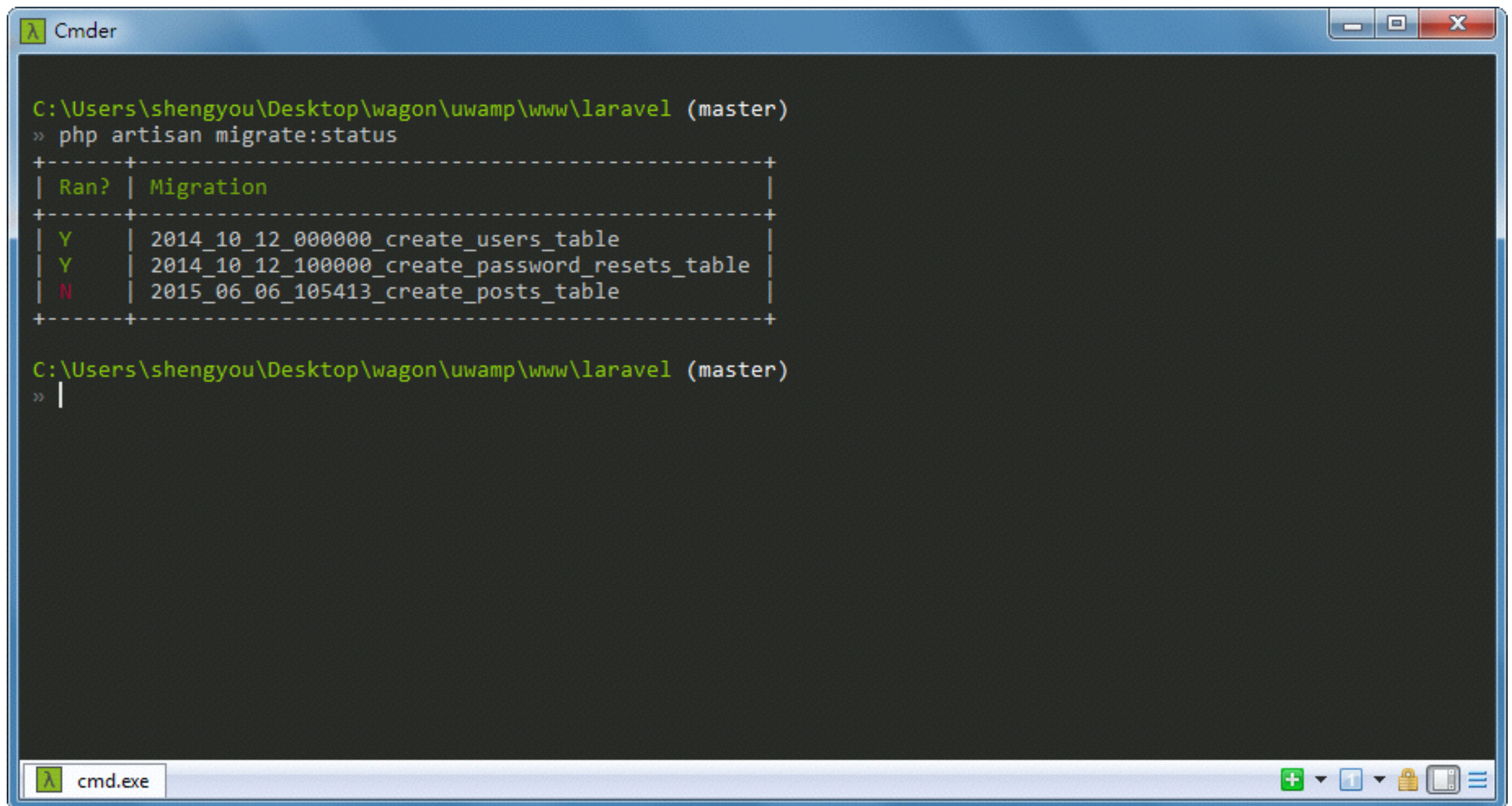
artisan migrate:status

- 透過 `artisan` 了解當前 `migrate` 執行狀況
 - `artisan` 會以表格的方式回傳目前所有 migration 檔被執行的狀態，從表格裡可以了解哪些 `migrate` 已經被執行、哪些還沒有？
- 範例：

```
$ php artisan migrate:status
```

執行 migrate:status

透過 artisan 了解 migrate 現況



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan migrate:status
+-----+-----+
| Ran? | Migration |
+-----+-----+
| Y    | 2014_10_12_000000_create_users_table |
| Y    | 2014_10_12_100000_create_password_resets_table |
| N    | 2015_06_06_105413_create_posts_table |
+-----+-----+

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

執行資料庫變更

- 撰寫完 migration 檔後，要透過 artisan 執行 migrate 指令，這些對資料庫的變更才真的會被執行

```
$ [php] artisan migrate
```

執行資料庫變更

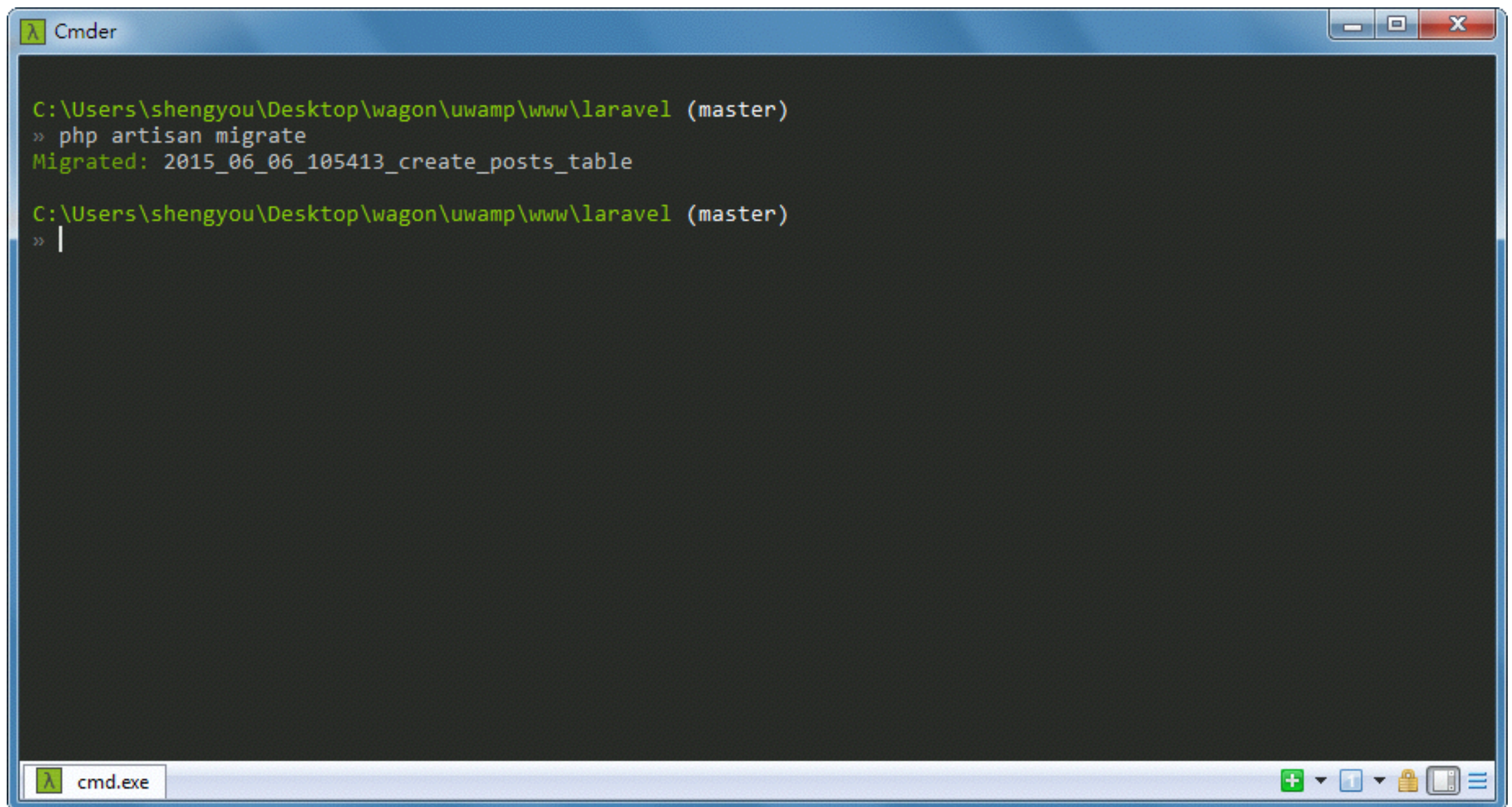
artisan migrate

- 呼叫 artisan 執行 migrate 動作
 - 撰寫完 migration 檔後，要執行此指令才會正式修改資料庫
 - artisan 會自動依照 migrations 資料表的紀錄進行版本控管
- 範例：

```
$ php artisan migrate
```

執行 migrate

透過 artisan 執行 migrate



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan migrate
Migrated: 2015_06_06_105413_create_posts_table

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

The screenshot shows a Windows Command Prompt window titled 'Cmder'. The command prompt is at the directory 'C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel' and is in '(master)' mode. The user has entered the command 'php artisan migrate', which has been executed successfully, resulting in the output 'Migrated: 2015_06_06_105413_create_posts_table'. The prompt is now waiting for another command, indicated by a vertical bar cursor.

檢查 DB 修改結果

選擇: migrations - localh x

localhost:8000/adminer/?server=localhost%3A3306&username=root&db=laravel_loc

語言: 繁體中文

MySQL » localhost:3306 » laravel_local » 選擇: migrations 登出

Adminer 4.2.1

DB: laravel_local

SQL命令 匯入 匯出
建立資料表

選擇 migrations
選擇 password_resets
選擇 posts
選擇 users

選擇資料 顯示結構 修改資料表 新增項目

選擇 搜尋 排序 限定 50 Text 長度 100 動作 選擇

SELECT * FROM `migrations` LIMIT 50 (0.001秒) 編輯

<input type="checkbox"/> Modify	migration	batch
<input type="checkbox"/> 編輯	2014_10_12_000000_create_users_table	1
<input type="checkbox"/> 編輯	2014_10_12_100000_create_password_resets_table	1
<input type="checkbox"/> 編輯	2015_06_06_105413_create_posts_table	2

(3行) ☐ 所有結果

Modify 儲存 Selected (0) 編輯 複製 刪除 匯出 (3) 匯入

Laravel 的 migrate 版次紀錄

新建的資料表

測試 rollback

- 雖然 Migration 會協助我們幫資料庫建立版本控制的回溯機制，但實際上資料庫變更裡的動作內容仍然是靠人類智慧去完成
- 因此，為確保版本控制的內容是無誤的，在將 Migration 送進版本控制之前，要先測試一下 **rollback** 的動作是不是正確無誤，以防止真的需要回溯時無法正確的回復到先前的狀態
- 千萬別偷懶不寫 **down** 動作，或是沒有測試 **rollback**！

```
$ [php] artisan migrate:rollback
```

回溯資料庫變更

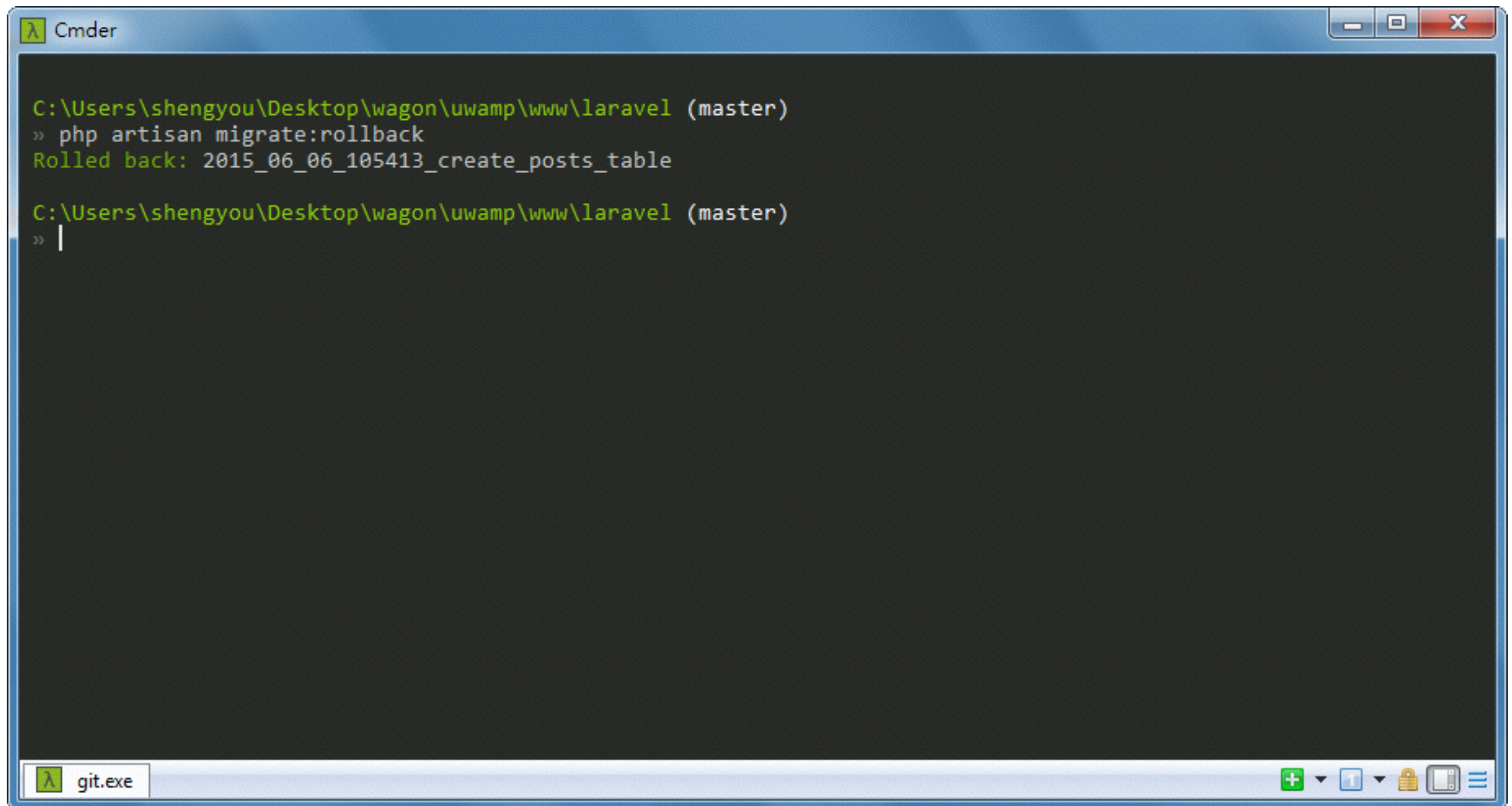
artisan migrate:rollback

- 將資料庫的內容回復到前一次的版本
 - artisan 會將上一次執行過的 migrate 動作透過 down 函式的內容回復到前一次的版本
 - artisan 會自動依照 migrations 資料表的紀錄進行版本控管
- 範例：

```
$ php artisan migrate:rollback
```

執行 migrate:rollback

回溯資料庫變更至上一個版本



```
C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel (master)
>> php artisan migrate:rollback
Rolled back: 2015_06_06_105413_create_posts_table

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

The screenshot shows a Cmder terminal window with a dark background and green text. The window title is 'Cmder'. The command prompt shows the current directory as 'C:\Users\shengyou\Desktop\wagon\uwamp\www\laravel' and the branch as '(master)'. The user has entered the command 'php artisan migrate:rollback', and the output is 'Rolled back: 2015_06_06_105413_create_posts_table'. The user has then entered another prompt character '>>' followed by a vertical bar '|', indicating the command is still running or waiting for input. At the bottom of the window, there is a taskbar with a 'git.exe' icon and several system icons on the right.

其他 migrate 指令

- 回到最初狀態

```
$ php artisan migrate:reset
```

- 回到最初後再重跑一次

```
$ php artisan migrate:refresh
```

- 了解 migrate 產生的 SQL 語法

```
$ php artisan migrate --pretend
```

小提醒 - 1

- 養成習慣使用 migration 工具來操作資料庫，絕對不要自己去動資料庫的任何設定，以免造成程式碼與實際資料庫內容不同步
- 除非有特別的原因，一個 migration 儘量只完成一個動作。需要對多個資料表變更時，就產生多個 migration 檔，讓每個 migrate 的動作單純

小提醒 - 2

- 每次執行 migration 時，都要測試一下 `migrate/migrate:rollback` (即 up/down 的內容) 無誤，確認動作內容都正確後，才將這個階段的變更放入版本控制內，並 `push` 出去提供給其他開發者同步
- 一旦將 migration 檔放入版本控制並 `push` 出去後，就不可以再修改 migration 檔裡的內容，以免造成其他人的不同步或錯誤。若因有誤而需要修改資料庫的狀況時，應該再建立新的 migration 來變更資料庫

小技巧 - 1

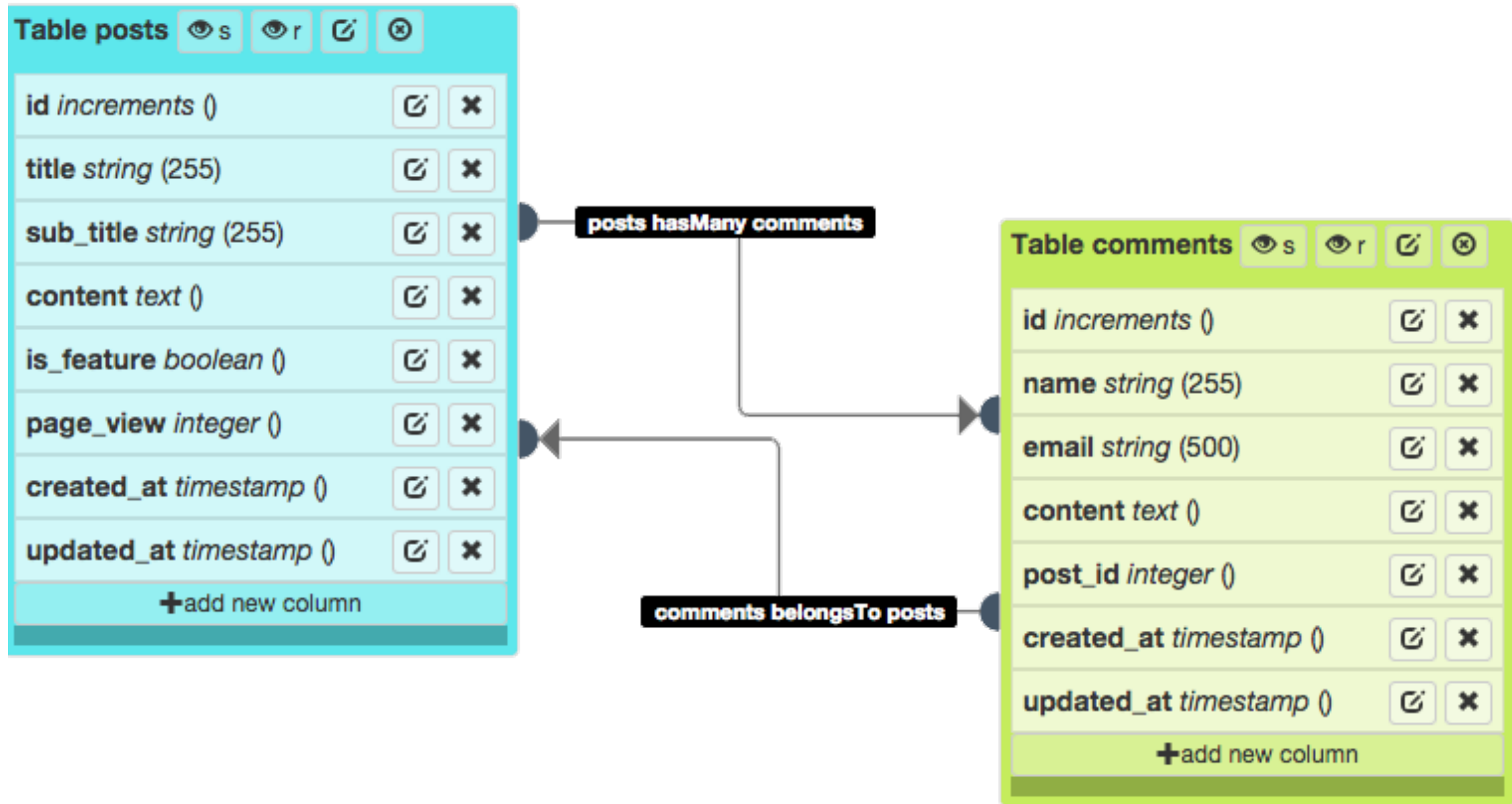
- 在輸入 migration 名稱時打錯字怎麼辦？
 - 先用 `artisan migrate:rollback`，重新命名 Class 名稱及檔案名稱，再執行 `composer dump-autoload` 及 `artisan migrate` 指令
 - 把產生出來的檔案手動刪除掉、資料庫內資料表砍掉，再重新產生一次

小技巧 - 2

- 不小心把 migrations 資料表弄壞、或是弄亂無法復原時怎麼辦？
- 先試試 `artisan migrate:reset`
- 把所有資料表全部手動刪除，再重新跑一次 `artisan migrate` 重建所有資料表

實作專案 Migration

專案資料庫設計



實作 migration

- 針對專案應用程式需要的三個資料表，用 `make:migration` 來產生三個 migration 檔
- 撰寫 migration 檔的內容
- 執行 `migrate`
- 在 GUI 裡檢查結果是否正確
- 測試 `migrate:rollback`

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？
 - Migration 的基本概念及使用 Migration 的好處
 - Laravel 的 make 相關指令
 - Laravel 的 migrate 相關指令
 - 設定專案資料庫及實作 Migration



歡迎提問討論