從學徒變大師

談 Laravel 框架擴充與套件開發

范聖佑 (Shengyou Fan)

PHPCon China 2016

2016/06/26

PHPCON
www.phpconchina.com

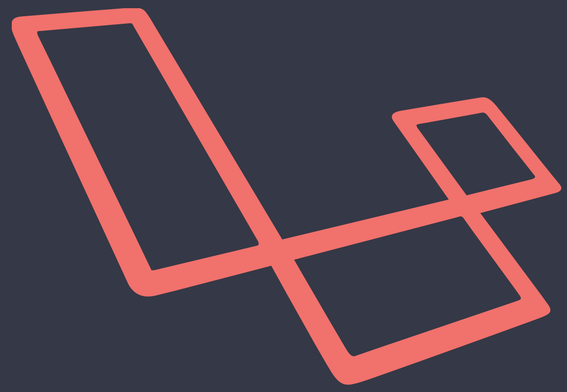
```
{
    "name": "shengyou/self-introduction",
    "description": "自我介紹",
    "authors": [
        {
            "name": "范聖佑 (Shengyou Fan)",
            "email": "shengyoufan@gmail.com",
            "homepage": "http://www.shengyoufan.com",
            "role": ["Researcher", "Evangelist"]
        }
    ],
    "support": {
        "facebook": "http://fb.me/shengyoufan",
        "twitter": "@shengyou",
        "wechat": "@shengyoufan"
    }
}
```

Page Script

5.3

CodeIgniter

v2.2



laravel

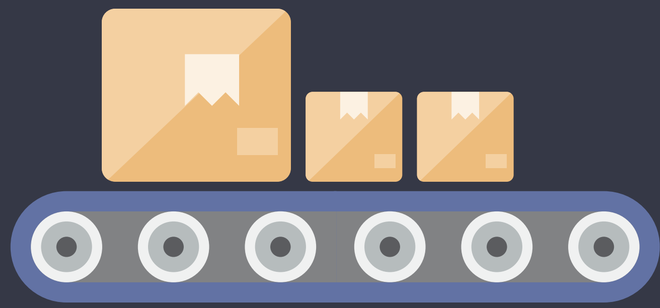
v4.0 ~ v5.2

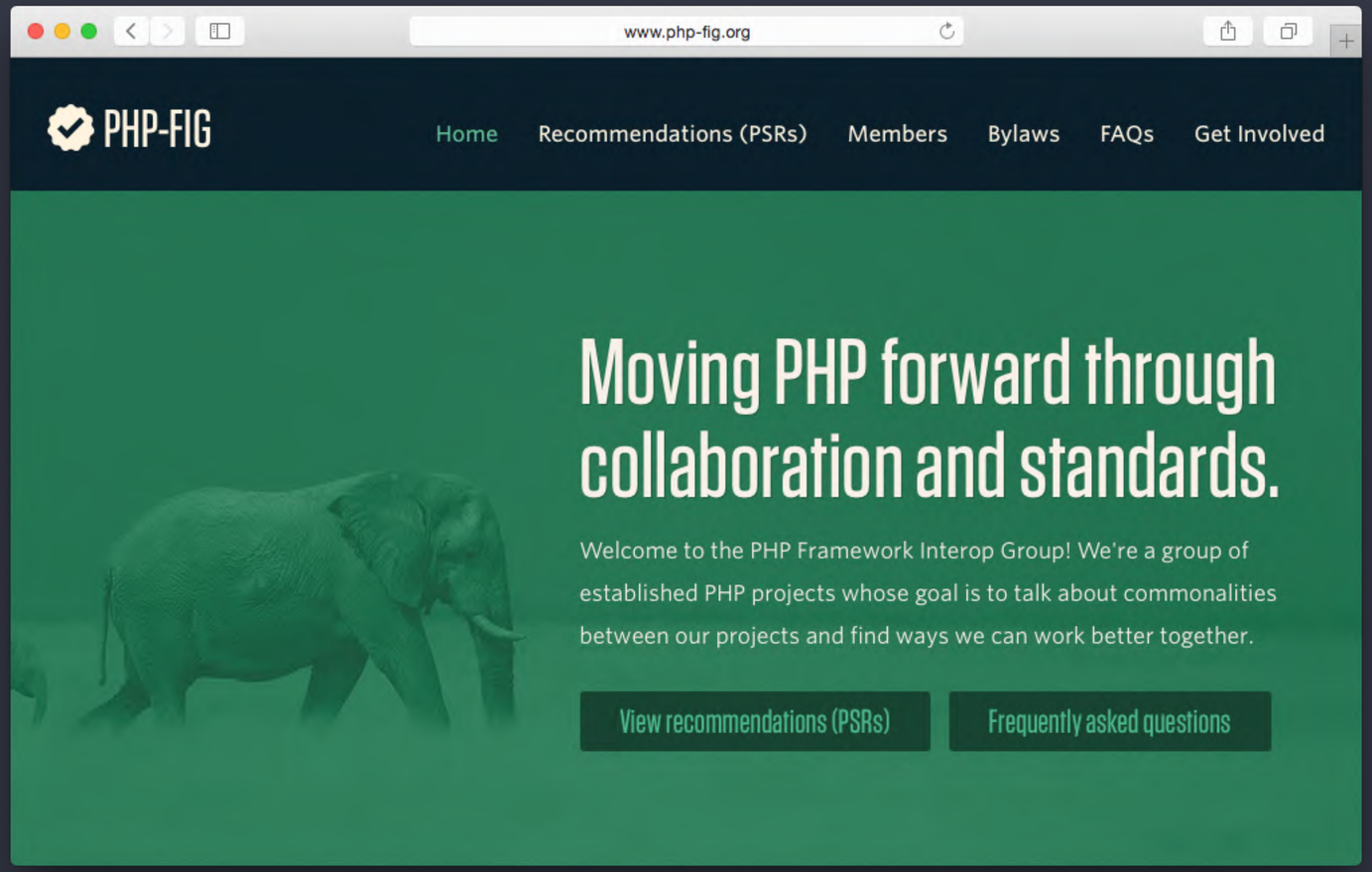
探索歷程

依賴 Composer

初始化、安裝、更新全靠它

尤其是 autoloading！

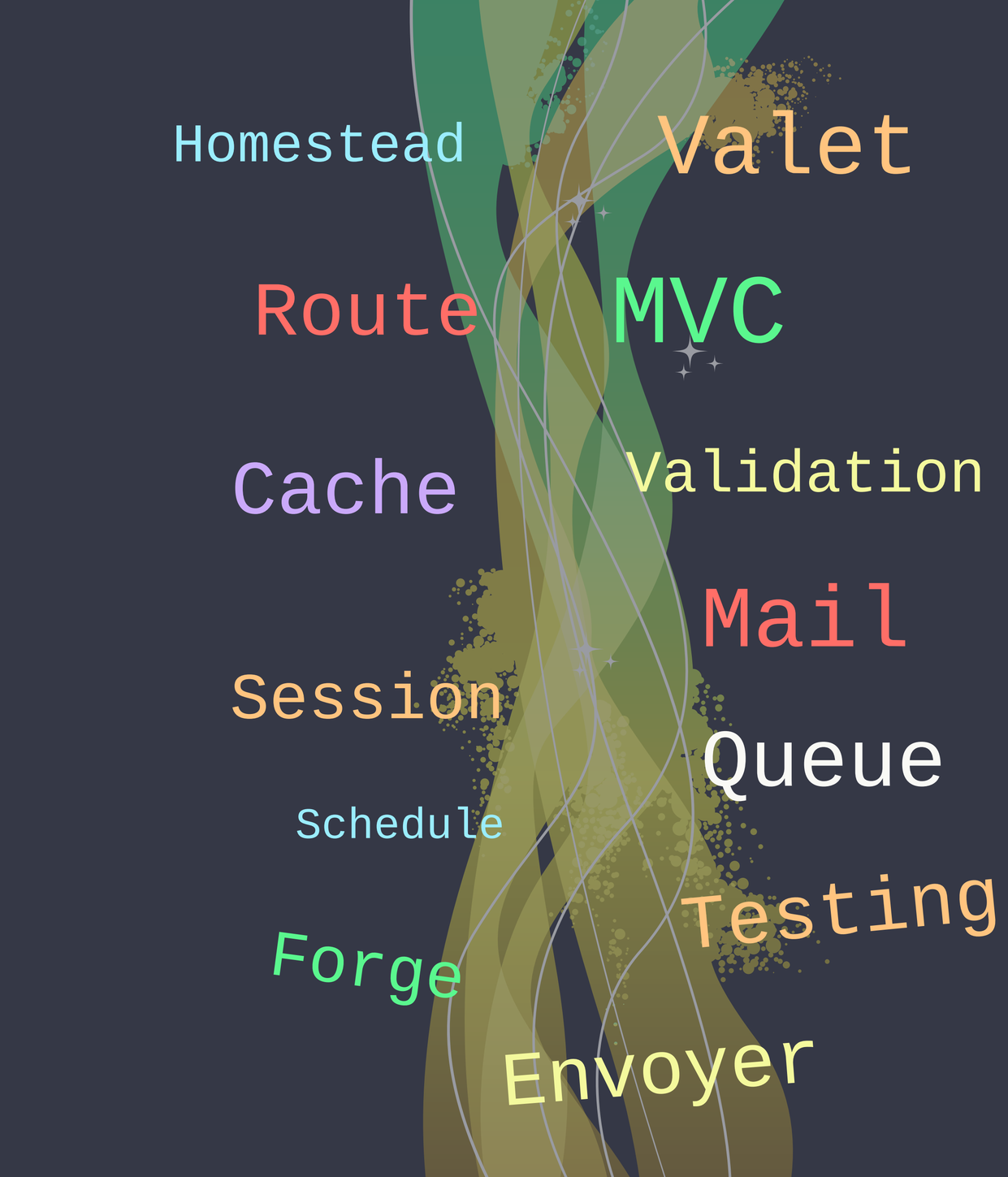




全面遵循 PSR

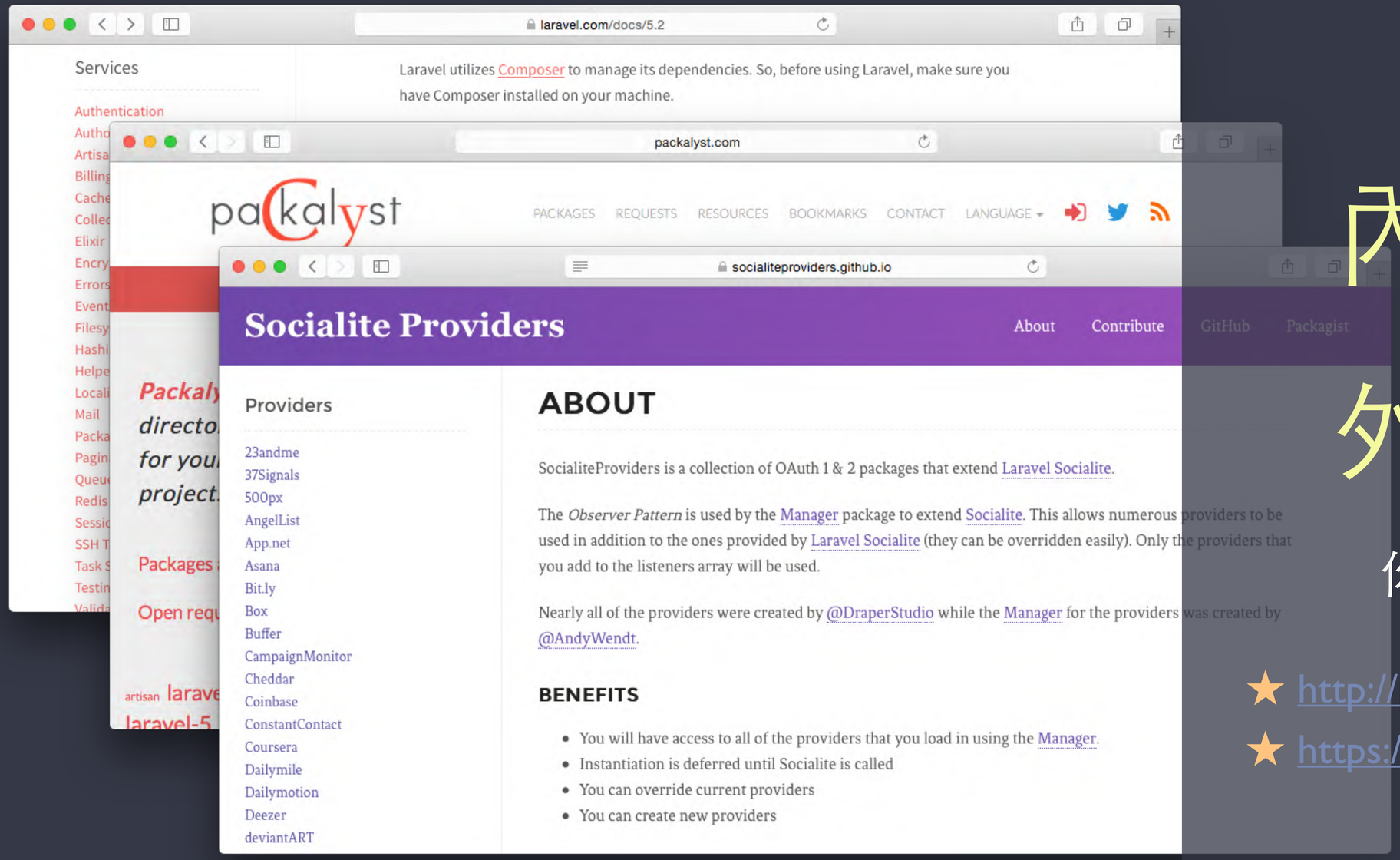
包括 PSR 1、PSR 2、PSR 4

★ 官方文件：https://laravel.com/docs/5.2/contributions#coding-style



不止是框架
更是完整流程

從開發環境、內建元件、測試部署
一條龍的完整框架



內建元件 + 外掛套件

你所需要的都有

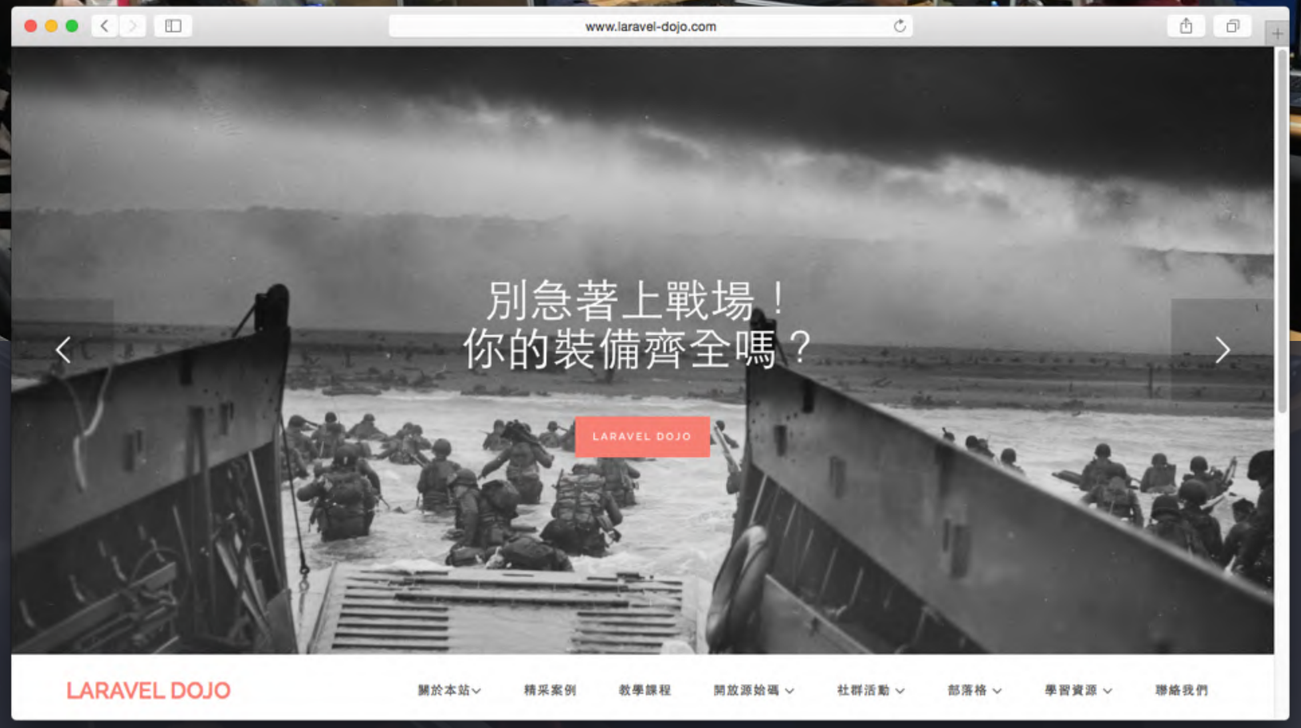
★ http://packalyst.com

★ https://socialiteproviders.github.io

Laravel 台灣

https://www.facebook.com/groups/laravel.tw

成立於 2013 現有 3600 位成員加入



Laravel 道場

http://www.laravel-dojo.com

致力於提供最好的 Laravel 教育訓練

2015.05
臺中科大資工系

2015.12
虎尾科大電算中心

2016.02
交通大學資服中心

2014.12
彰師大資工系

2015.07
新北市樹林國小

2016.01
中華電信學院

2016.03
正修科大資工系

推廣了近三年，

最常聽到的回應是...

Laravel 看起來
好像不錯

But…

But

但

但

但…

…

但…

But…

但

導入框架的疑慮

會不會被框架綁死？

框架會持續維護嗎？

有辦法跟現有專案整合嗎？

有辦法擴充嗎？

會不會很難改啊？

Laravel: From Apprentice To Artisan

從學徒變大師

框架作者 Taylor Otwell 親自撰寫
說明核心架構及框架擴充

> https://leanpub.com/laravel

核心機制

範例情境

- 使用者在網站上註冊，註冊完會發一封通知簡訊
- 使用 Nexmo 簡訊寄送服務

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register()
    {
        // ...

        // 寄送簡訊



        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register()
    {
        // ...

        // 寄送簡訊
        $nexmoSender = new NexmoSender();


        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register()
    {
        // ...

        // 寄送簡訊
        $nexmoSender = new NexmoSender();
        $nexmoSender->send('{PHONE}', '{MESSAGE}');

        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register()
    {
        // ...

        // 寄送簡訊
        $nexmoSender = new NexmoSender();
        $nexmoSender->send('{PHONE}', '{MESSAGE}');

        // ...
    }
}
```

NexmoSender 和 UsersController 綁得太緊

Dependency Injection

依頼注入

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{


    public function __construct()
    {

    }

    public function register()
    {
        // ...
        $nexmoSender = new NexmoSender();
        $nexmoSender->send('{PHONE}', ‘{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{


    public function __construct(NexmoSender $smsSender)
    {

    }

    public function register()
    {
        // ...
        $nexmoSender = new NexmoSender();
        $nexmoSender->send('{PHONE}', ‘{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register()
    {
        // ...
        $nexmoSender = new NexmoSender();
        $nexmoSender->send('{PHONE}', ‘{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register()
    {
        // ...
        $nexmoSender = new NexmoSender();
        $nexmoSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register()
    {
        // ...
        $nexmoSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register()
    {
        // ...
        $this->smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

覺得很囉嗦？

Laravel 有一招更簡單的！

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register()
    {
        // ...
        $this->smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register(NexmoSender $smsSender)
    {
        // ...
        $this->smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register(NexmoSender $smsSender)
    {
        // ...
        $this->smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    protected $smsSender;

    public function __construct(NexmoSender $smsSender)
    {
        $this->smsSender = $smsSender;
    }

    public function register(NexmoSender $smsSender)
    {
        // ...
        $smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(NexmoSender $smsSender)
     {
         // ...
         $smsSender->send('{PHONE}', '{MESSAGE}');
         // ...
     }
}
```

怎麼辦到的？

IoC Container

IoC 容器

Laravel 的 Service Container

1. 目前的 type hinting 有註冊任何綁定嗎？

```php
class UsersController extends Controller
{
    public function register(NexmoSender $smsSender)
    {
        // ...
    }
}
```

```
NexmoSender::class => ?
```

Laravel 的 Service Container

1. 目前的 type hinting 有註冊任何綁定嗎？
2. 沒有？試著 Reflect 這個 Class，以及其相依 (遞迴)

```php
class UsersController extends Controller
{
    public function register(NexmoSender $smsSender)
    {
        // ...
    }
}
```

```php
new RefectionClass('NexmoSender');
new NexmoSender(params);
```

狀況題：需求變更

- 業務需求變更，要換簡訊服務商
- 改用 Twilio 簡訊寄送服務

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(NexmoSender $smsSender)
    {
        // ...
        $smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(TwilioSender $smsSender)
    {
        // ...
        $smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(TwilioSender $smsSender)
    {
        // ...
        $smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(TwilioSender $smsSender)
    {
        // ...
        $smsSender->phone('{PHONE}');
        $smsSender->message('{MESSAGE}');
        $smsSender->call();
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(TwilioSender $smsSender)
    {
        // ...
        $smsSender->phone('{PHONE}');
        $smsSender->message('{MESSAGE}');
        $smsSender->call();
        // ...
    }
}
```

只要每次有需求變更，就得修改對應的程式碼！

透過建立 Interface
讓寄送簡訊的動作變得一致

建立契約

1. 將寄發簡訊的動作抽象成一個 Interface

```php
// app/Contracts/SmsSenderContract.php
interface SmsSenderContract
{
    public function send($phoneNumber, $message);
}
```

宣告 SmsSenderContract 並定義 methods

建立契約

1. 將寄發簡訊的動作抽象成一個 Interface
2. 寄送簡訊的 Class 都要實作 Interface

```php
// app/Libraries/NexmoSender.php
class NexmoSender
{




}

// app/Libraries/TwilioSender.php
class TwilioSender
{




}
```

```php
// app/Libraries/NexmoSender.php
class NexmoSender implements SmsSenderContract
{
    public function send($phoneNumber, $message)
    {
        // ...
    }
}

// app/Libraries/TwilioSender.php
class TwilioSender implements SmsSenderContract
{
    public function send($phoneNumber, $message)
    {
        // ...
    }
}
```

Sender 實作 SmsSenderContract 及其 methods

建立契約

1. 將寄發簡訊的動作抽象成一個 Interface
2. 寄送簡訊的 Class 都要實作 Interface
3. Controller 的 type hinting 改成 Interface

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(TwilioSender $smsSender)
    {
        // ...
        $smsSender->phone('{PHONE}');
        $smsSender->message('{MESSAGE}');
        $smsSender->call();
        // ...
    }
}
```

修改 type hinting

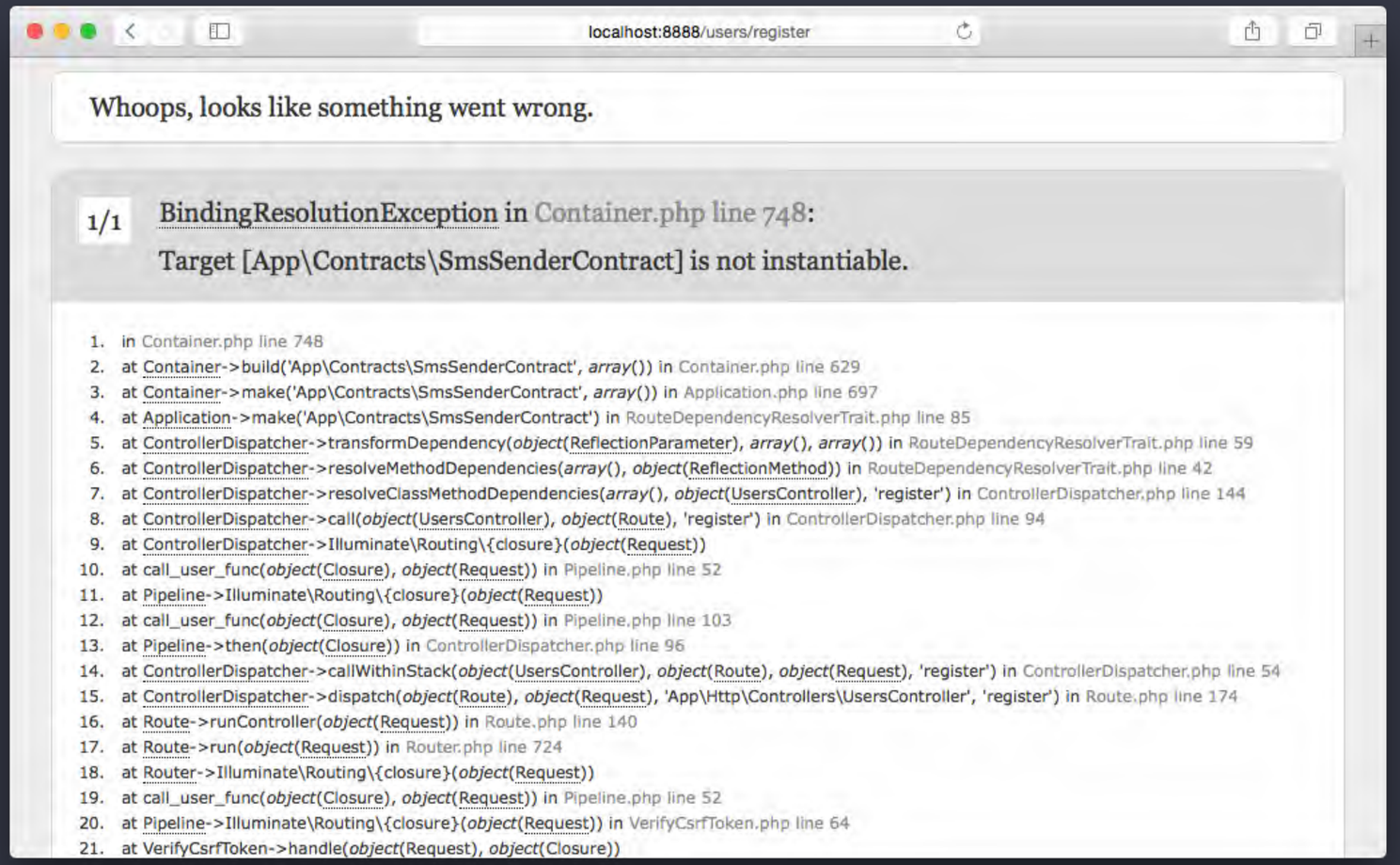
```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(SmsSenderContract $smsSender)
    {
        // ...
        $smsSender->phone('{PHONE}');
        $smsSender->message('{MESSAGE}');
        $smsSender->call();
        // ...
    }
}
```

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(SmsSenderContract $smsSender)
    {
        // ...
        $smsSender->phone('{PHONE}');
        $smsSender->message('{MESSAGE}');
        $smsSender->call();
        // ...
    }
}
```

改用 Interface 定義的 methods

```php
// app/Http/Controllers/UsersController.php
class UsersController extends Controller
{
    public function register(SmsSenderContract $smsSender)
    {
        // ...
        $smsSender->send('{PHONE}', '{MESSAGE}');
        // ...
    }
}
```

照著這樣的架構寫
結果出錯了？



因為 Interface 還未綁定 Class

Service Provider

服務提供者

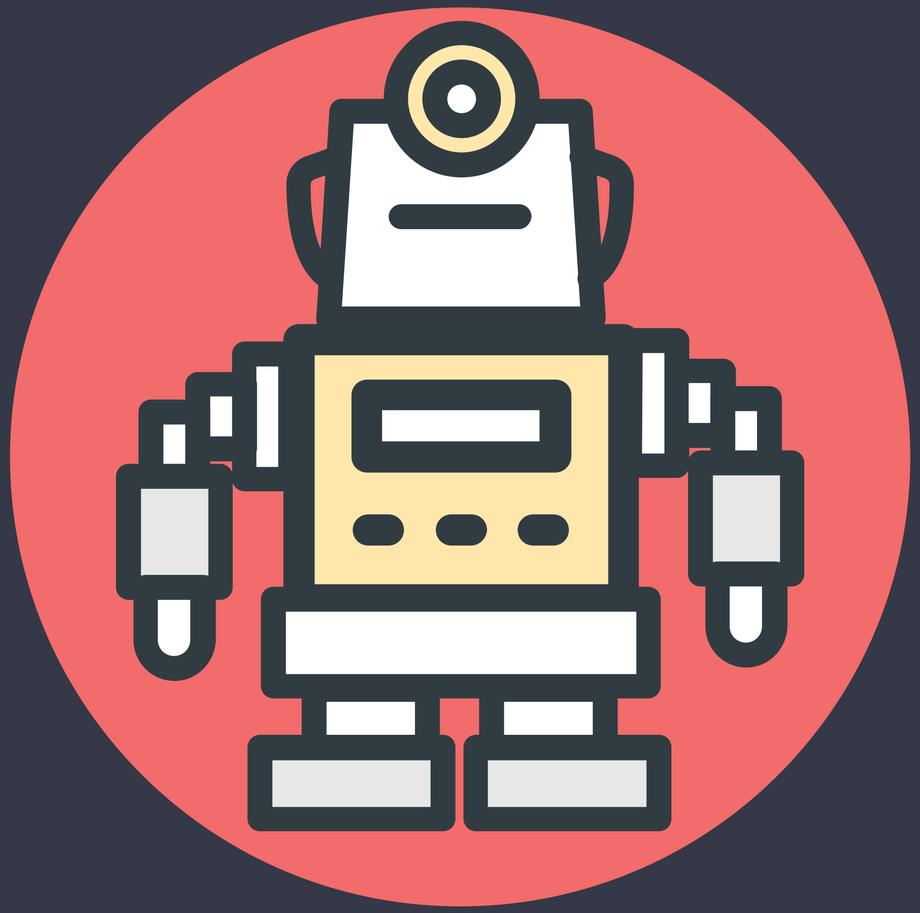
由 Service Provider 組合而成
的應用程式架構

Client

index.php

Laravel

initializing

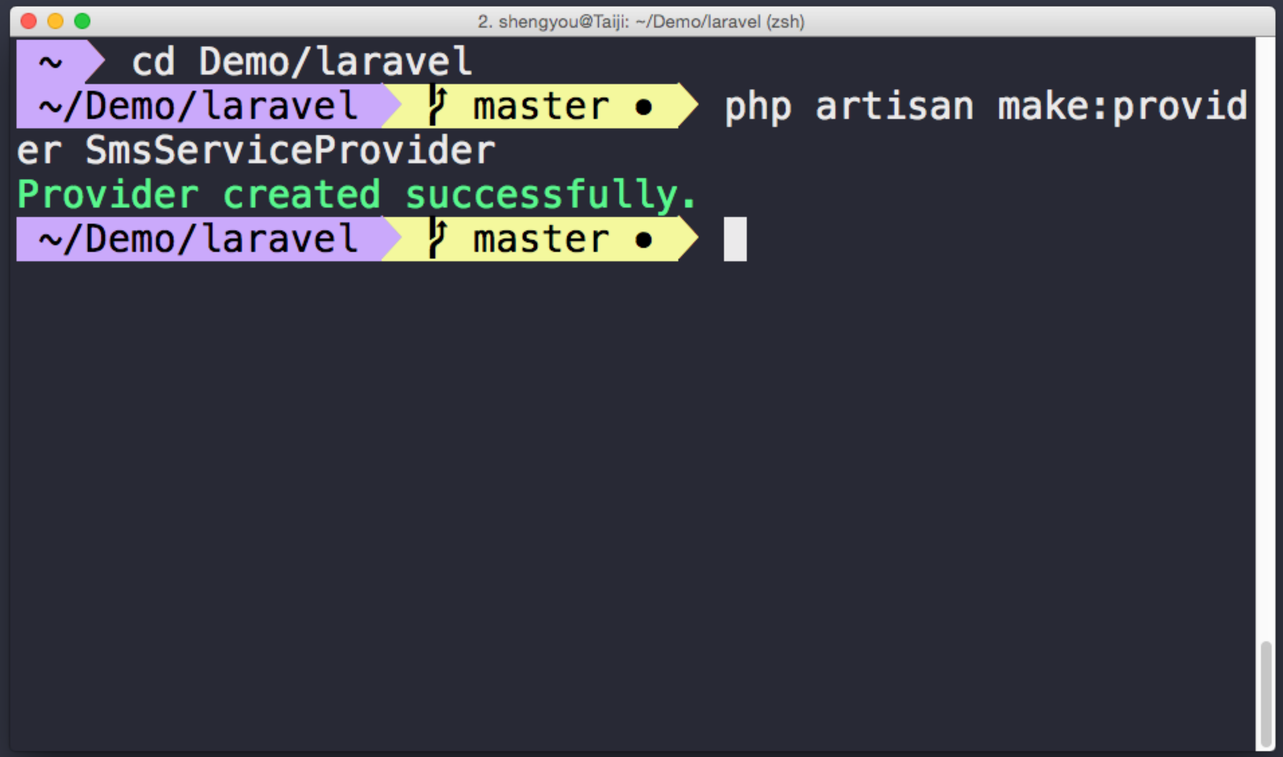


```php
// config/app.php
'providers' => [
    /*
     * Laravel Framework Service Providers...
     */
    // ...
    Illuminate\Mail\MailServiceProvider::class,
    Illuminate\Queue\QueueServiceProvider::class,
    Illuminate\View\ViewServiceProvider::class,

    /*
     * Application Service Providers...
     */
    App\Providers\AppServiceProvider::class,
    App\Providers\AuthServiceProvider::class,
    App\Providers\EventServiceProvider::class,
    App\Providers\RouteServiceProvider::class,
],
```

實作 SmsServiceProvider

1. 建立 SmsServiceProvider.php



$ php artisan make:provider SmsServiceProvider

實作 SmsServiceProvider

1. 建立 SmsServiceProvider.php
2. 在 SmsServiceProvider 裡註冊綁定、解決相依

```php
// app/Providers/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {



    }
}
```

```php
// app/Providers/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(

        );
    }
}
```

```php
// app/Providers/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(
            SmsSenderContract::class
        );
    }
}
```

Interface 定義

```php
// app/Providers/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(
            SmsSenderContract::class, NexmoSender::class
        );
    }
}
```

Interface 定義　　綁定的 Class

Class 的相依參數怎麼處理？

```php
// app/Providers/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(
            SmsSenderContract::class, NexmoSender::class
        );
    }
}
```

```php
// app/Providers/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(
            SmsSenderContract::class, function() {
            return new NexmoSender('{API_KEY}', '{API_SECRET}');
        });
    }
}
```

第二個參數改成 Closure，在 Closure 內處理

實作 SmsServiceProvider

1. 建立 SmsServiceProvider.php
2. 在 SmsServiceProvider 裡註冊綁定、解決相依
3. 在 app.php 裡啟動 SmsServiceProvider

```php
// config/app.php
'providers' => [

    /*
     * Laravel Framework Service Providers...
     */
    // ...
    Illuminate\Session\SessionServiceProvider::class,

    App\Providers\SmsServiceProvider::class,

    /*
     * Application Service Providers...
     */
    App\Providers\AppServiceProvider::class,
    // ...

],
```

沒問題！
包成套件給你

聽說你寫了發送簡訊的功能，
我的專案也能用嗎？

擴充套件

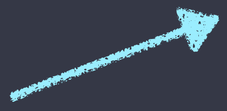
發佈套件

1. 建立 Composer 套件結構

Laravel 專案

SMS Package

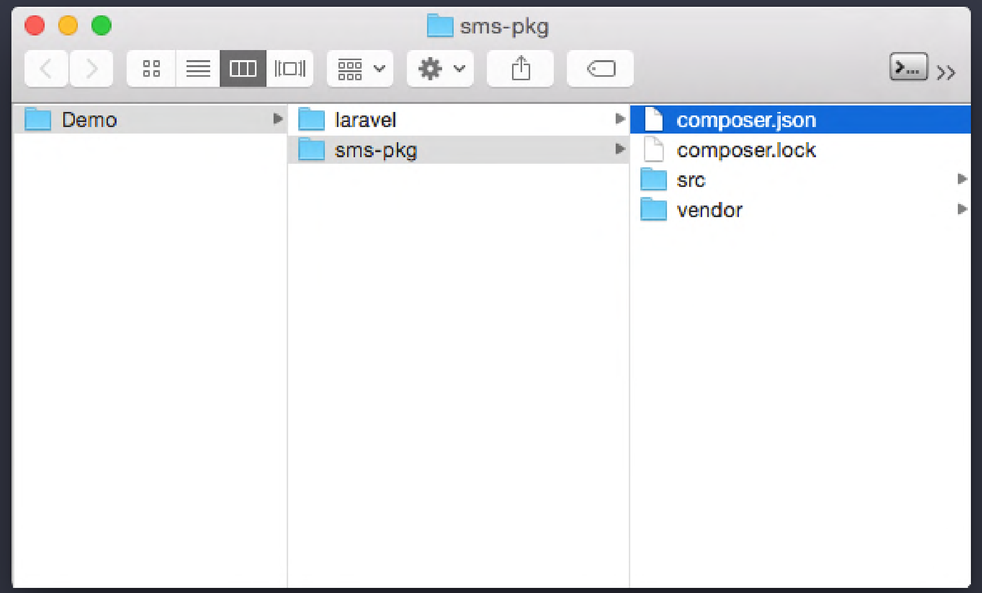
- laravel
  - app
    - Contracts
      - SmsSenderContract.php
    - Libraries
      - NexmoSender.php
    - Providers
      - SmsServiceProvider.php



- sms-pkg
  - src
    - Contracts
      - SmsSenderContract.php
    - Senders
      - NexmoSender.php
    - SmsServiceProvider.php
  - composer.json

發佈套件

1. 建立 Composer 套件結構
2. 設定套件的 `composer.json`



SMS Package

```
// composer.json
{
    "name": "vendor/sms-pkg",
    "require": {
        "php": ">=5.5.9",
        "illuminate/support": "5.2.*"
    },
    "autoload": {
        "psr-4": {
            "MyNamespace\\Sms\\": "src"
        }
    }
}
```

發佈套件

1. 建立 Composer 套件結構
2. 設定套件的 `composer.json`
3. 選擇發佈套件的方式

發佈管道

- 公開發佈
  - 將程式碼放置在 Github 上
  - 在 Packagist 上註冊套件資訊
    - ★ 參考：http://oomusou.io/laravel/laravel-package-hello-world/
- 私密使用
  - 將程式碼放置在私有版本控制系統上
  - 架設 Satis 或 Toran Proxy
    - ★ 參考：http://www.slideshare.net/shengyou/modern-web-conf-2015-php-composer

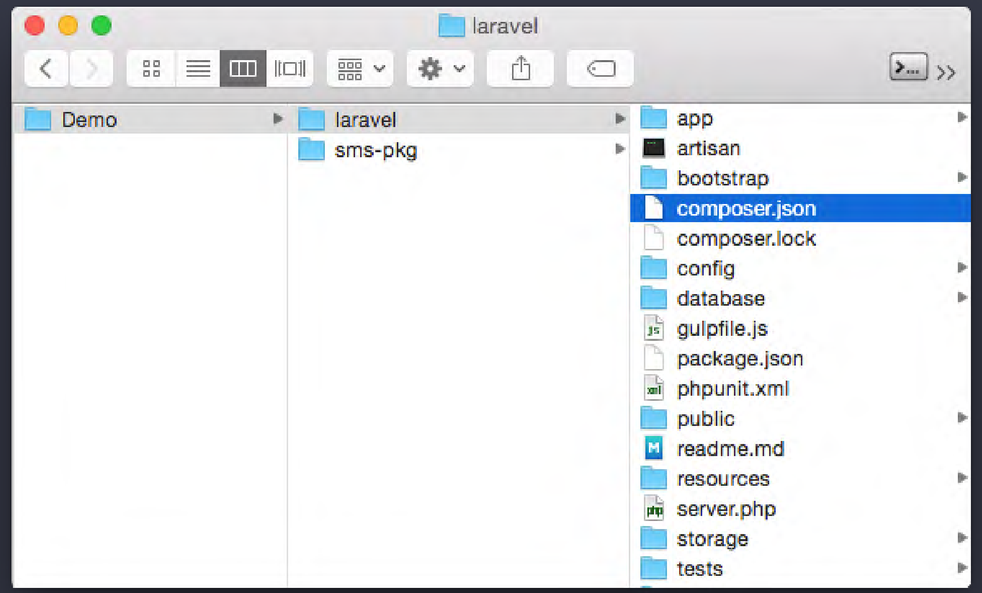
搞定！

可以使用了

怎麼安裝？

安裝套件

1. 設定專案的 composer.json



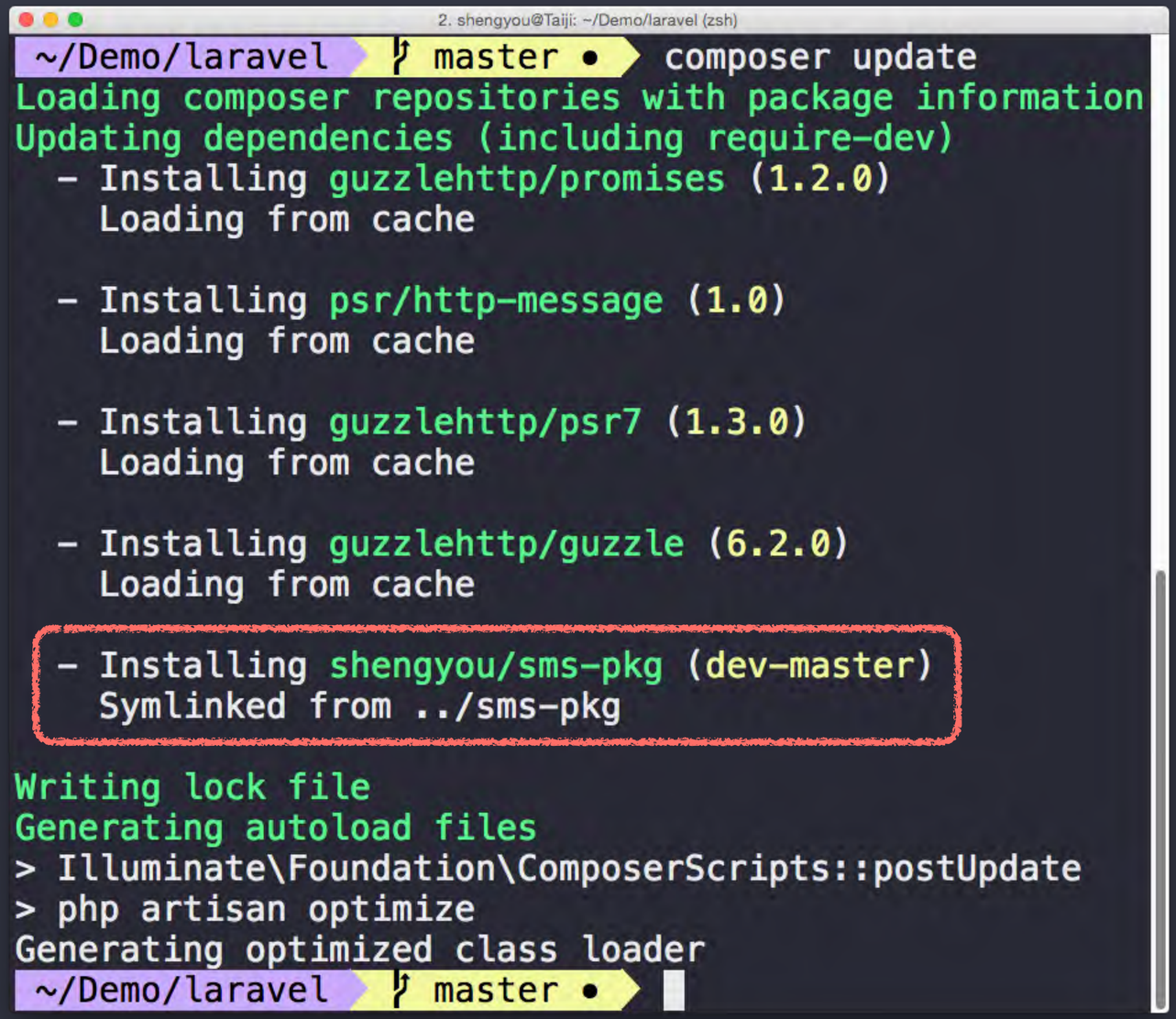
Laravel 專案

```
// composer.json
{
    "name": "vendor/myproject",
    "require": {
        "php": ">=5.5.9",
        "laravel/framework": "5.2.*",
        "vendor/sms-pkg": "dev-master"
    },
    "repositories": [
        {
            "type": "path",
            "url": "../sms-pkg"
        }
    ],
}
```

依 repositories 現況設定 type 及 url

安裝套件

1. 設定專案的 composer.json
2. 使用 Composer 更新相依套件



$ composer update

安裝套件

1. 設定專案的 composer.json
2. 使用 Composer 更新相依套件
3. 設定專案 config/app.php 內的 providers 陣列

```php
// config/app.php
'providers' => [

    /*
     * Laravel Framework Service Providers...
     */
    //...
    Illuminate\Session\SessionServiceProvider::class,

    MyNamespace\Providers\SmsServiceProvider::class,

    /*
     * Application Service Providers...
     */
    App\Providers\AppServiceProvider::class,
    // ...

],
```

Ta-da!

讚啦！

也給我一份！

主題總結

今日討論主題

特色綜覽

依賴 Composer

全面遵循 PSR

一條龍方案

完整生態系

核心機制

Dependency Injection

IoC Container

Service Provider

套件開發

套件結構

設定 composer.json

發佈套件

安裝套件

Too much Java?

少一點套路、多一點真誠？

隨機應變

一切都跟需求有關

更容易了解
別人的套件原理

好處 1

非同步開發
好測試、好分工

好處 2

```php
// app/Libraries/DummySender.php
class DummySender implements SmsSenderContract
{
    public function send()
    {
        // 模擬或假裝有送出簡訊
    }
}
```

```php
// app/Services/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(
            SmsSenderContract::class, function() {
            return new NexmoSender('{API_KEY}', '{API_SECRET}');
        });
    }
}
```

```php
// app/Services/SmsServiceProvider.php
class SmsServiceProvider extends ServiceProvider
{
    public function register()
    {
        $this->app->bind(
            SmsSenderContract::class, function() {
            return new DummySender();
        });
    }
}
```

還沒正式接上簡訊系統前
先做一個假的讓程式可以運作

```php
// tests/ExampleTest.php
class ExampleTest extends TestCase
{
    public function testUsersControllerSendSms()
    {
        // Arrange
        $smsSender = Mockery::mock(SmsSenderContract::class);
        $smsSender->shouldReceive('send')->once(); // Assert
        app()->instance(SmsSenderContract::class, $smsSender);

        // Act
        $this->action('POST', 'UsersController@register');
    }
}
```

使用 Interface 好 Mock、好測試

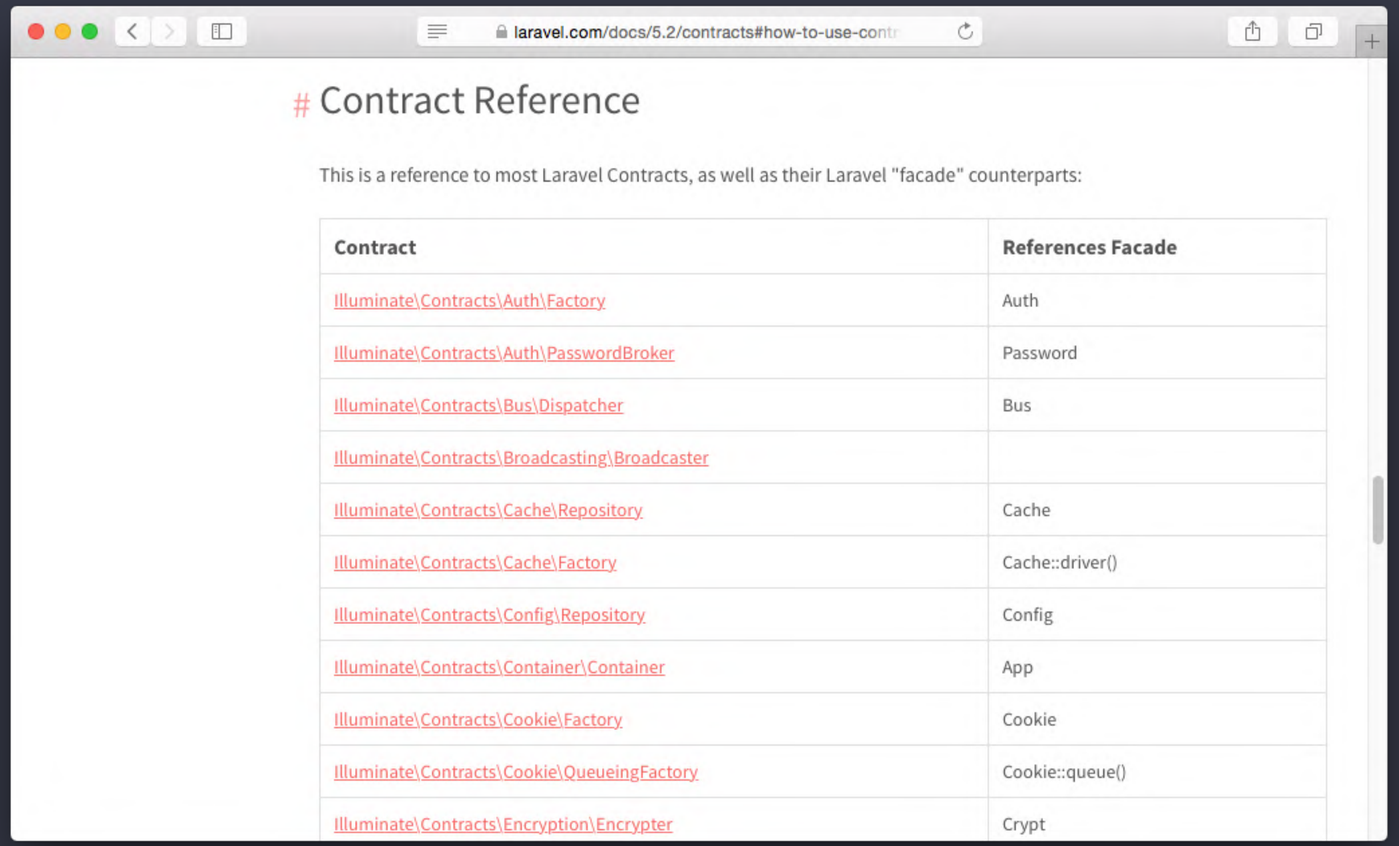
隨心所欲
抽換內建元件

好處 3

一切照合約走

Laravel Contracts

要抽換元件時，只要該元件有實作 Interface，在 IoC 裡抽換綁定就可以了



★ 官方文件：https://laravel.com/docs/5.2/contracts#how-to-use-contracts

更換樣板引擎？

- `ytake/laravel-smarty`
  - 將 Smarty 樣板引擎整合至 Laravel
    ★ 參考：https://github.com/ytake/Laravel.Smarty
- `rcrowe/twigbridge`
  - 將 Twig 樣板引擎整合至 Laravel
    ★ 參考：https://github.com/rcrowe/TwigBridge

工商服務

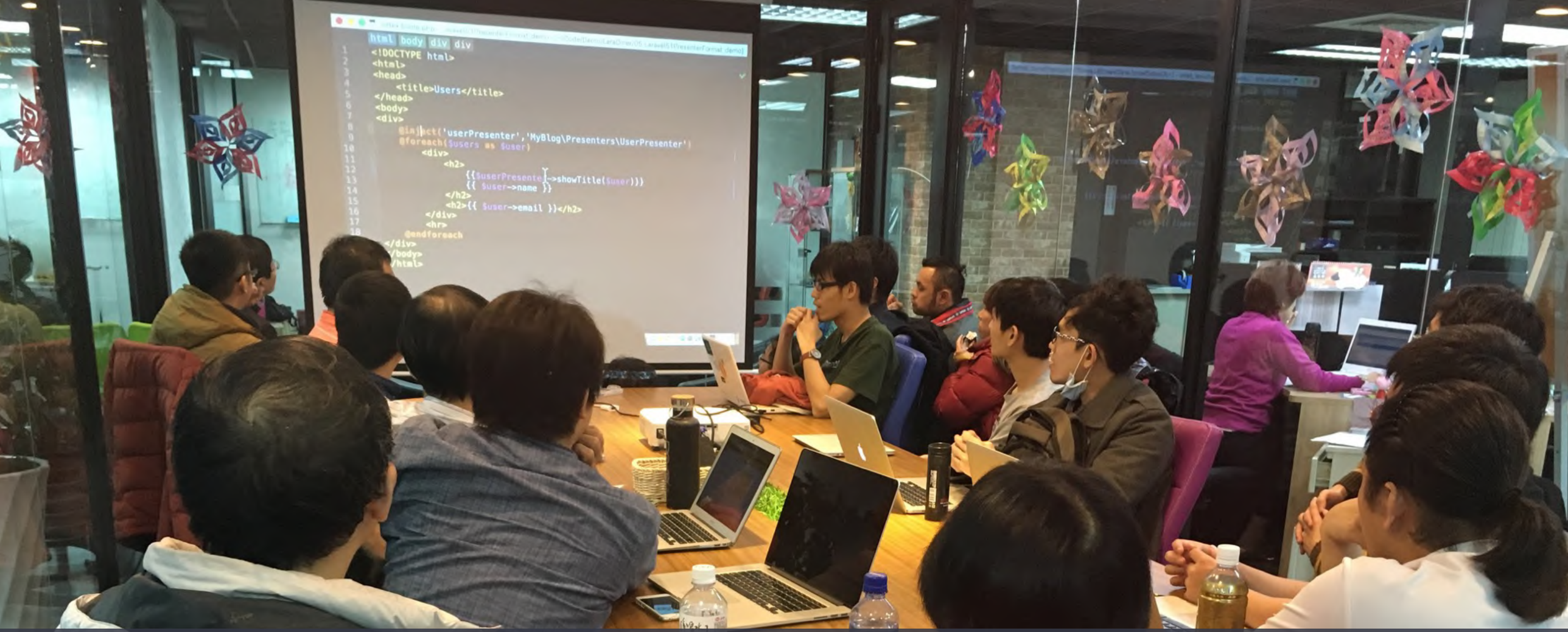
歡迎大家來台灣玩！

PHPConf Taiwan 2016

年度 PHP 盛會，將於 2016/10/29 舉辦

PHP 也有 Day

每月舉辦一次的 PHP 講座型聚會



Laradiner

每月舉辦一次的 Laravel 讀書會

Laradebut

每月舉辦一次的 Laravel 新手村

Vuediner

與友團合辦的Vue.js 讀書會

Q & A

感謝聆聽 · 歡迎交流