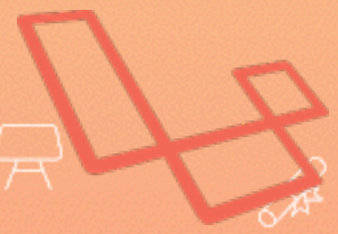
適用 5.1 版



View 與 Blade 樣板引擎

范聖佑 Shengyou Fan

新北市樹林國小 (2015/07/08)

```php
/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return Response
 */
public function show($id)
{
    $post = Post::with('comments')->where('id', $id)->first();

    return View::make('post.show')->with('post', $post);
}
```

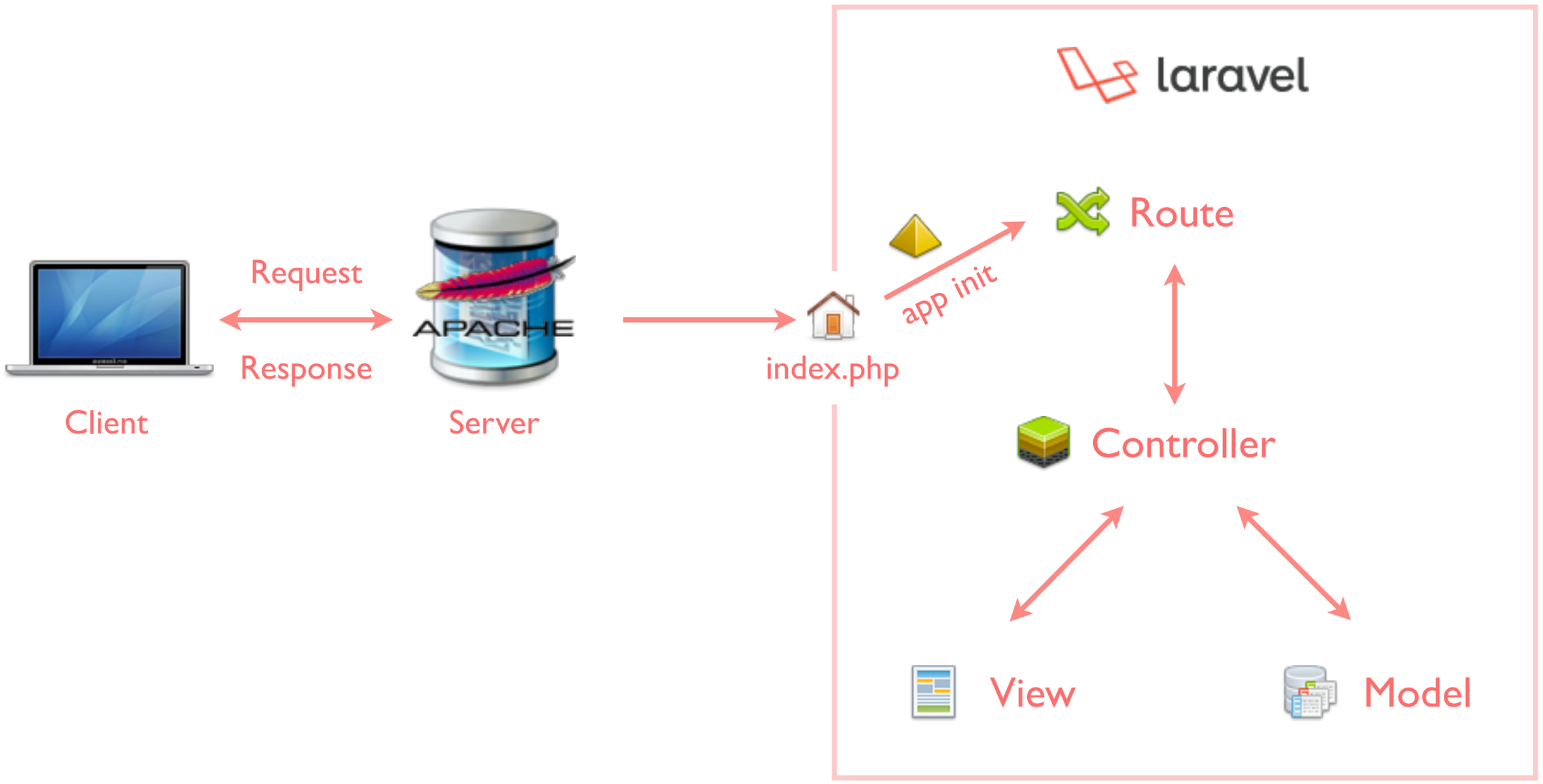
單元主題

- MVC 架構是什麼？什麼是 View？
- 使用 View 與 樣板引擎在開發上的好處
- 學習 Blade 樣板語法
- 示範如何從網路上下載樣板並使用 Laravel 樣板引擎整合進實作專案前台內

MVC 裡的 View

★ 參考文件：http://laravel.com/docs/5.1/views

Laravel 的 MVC 分工



★ 參考說明：http://laravel-recipes.com/recipes/52/understanding-the-request-lifecycle

沒有導入 MVC 的話...

- 傳統直譯式的 PHP 寫法，往往將程式運作邏輯和 HTML 混雜在一起，也就是所謂義大利麵式程式碼
- 這樣的寫法除了難以閱讀外，更難以維護...

```
<?php
// 連線資料庫
$mysqli = new mysqli('localhost', 'root', 'root', 'demo');
/* 中間略 */
?>
<html>
<!-- 中間略 -->
<ul>
  <?php $query = "SELECT * FROM posts ORDER BY..."; ?>
  <?php foreach($mysqli->query($query) as $row): ?>
  <li><?=$row['title']?></li>
  <?php endforeach; ?>
</ul>
```

什麼是 MVC 的 View？

- 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，View 的功能就是負責儲存 HTML，並將要顯示動態資料的部份留下來，等待被呼叫的時候由外部傳入
- 如此即可讓商業邏輯 (M)、顯示 (V) 與程式控制 (C) 三者獨立開來，彼此分工、互相合作

Laravel 裡的 View

- Laravel 本身就是一個以 MVC 為基底設計的框架，所有的 View 都統一放在 resources/views 資料夾底下
- 所有 View 的檔名皆以 *.php 做為結尾，檔案內容就是一般的 HTML 程式碼
- 當要顯示對應的頁面時，只要透過 view() 這個 helper function 就可以將內容從 Route 回傳至使用者端

```
Route::get('{uri}', function()
{
    return view('{path.view_file_name}');
});
```

先定義 View 的內容

- 先將 HTML 寫在 resources/views/ 底下
- 所有 View 的副檔名皆為 *.php
- 若有分類的需求，可任意開子資料夾整理

```
// resources/views/home/index.php
<!doctype html>
<html>
    <head><!-- 略 --></head>
    <body>
        <h1>Home</h1>
    </body>
</html>
```

直接從 Route 回傳

- 在這個階段，先直接從 Route 將 View 回傳至使用者端 (後續會再應用 Controller、Model)
- 在 Route 裡 使用 `view()` 這個 helper function 直接回傳，特別注意路徑要設定正確，中間資料夾可用「.」區隔，且不需要加副檔名 (*.php)

```
// app/Http/routes.php
Route::get('/', function()
{
    return view('home.index');
});
```

引入様板

下載樣板

- 在本工作坊裡，我們將使用 Start Bootstrap 的樣板來快速建置我們需要的所有頁面，請參考講義內容下載樣板原始檔：
  - Clean Blog：http://startbootstrap.com/template-overviews/clean-blog/
  - Blog Post：http://startbootstrap.com/template-overviews/blog-post/

★ P.S 當然！您可以使用自己喜歡的樣板

整合樣板至框架內

- 把樣板的 assets 放到 public 資料夾內
- 把樣板內的 *.html 依需求命名成 *.php 放到 resources/views 底下
- 移除預設的 resources/views/welcome.blade.php
- 更新 app/Http/routes.php 內的設定成回傳 view
- 修正連結路徑 (頁面內 css、js、img 的路徑)
- 確認每一個 route 回傳的頁面內容

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

Blade 樣板引擎

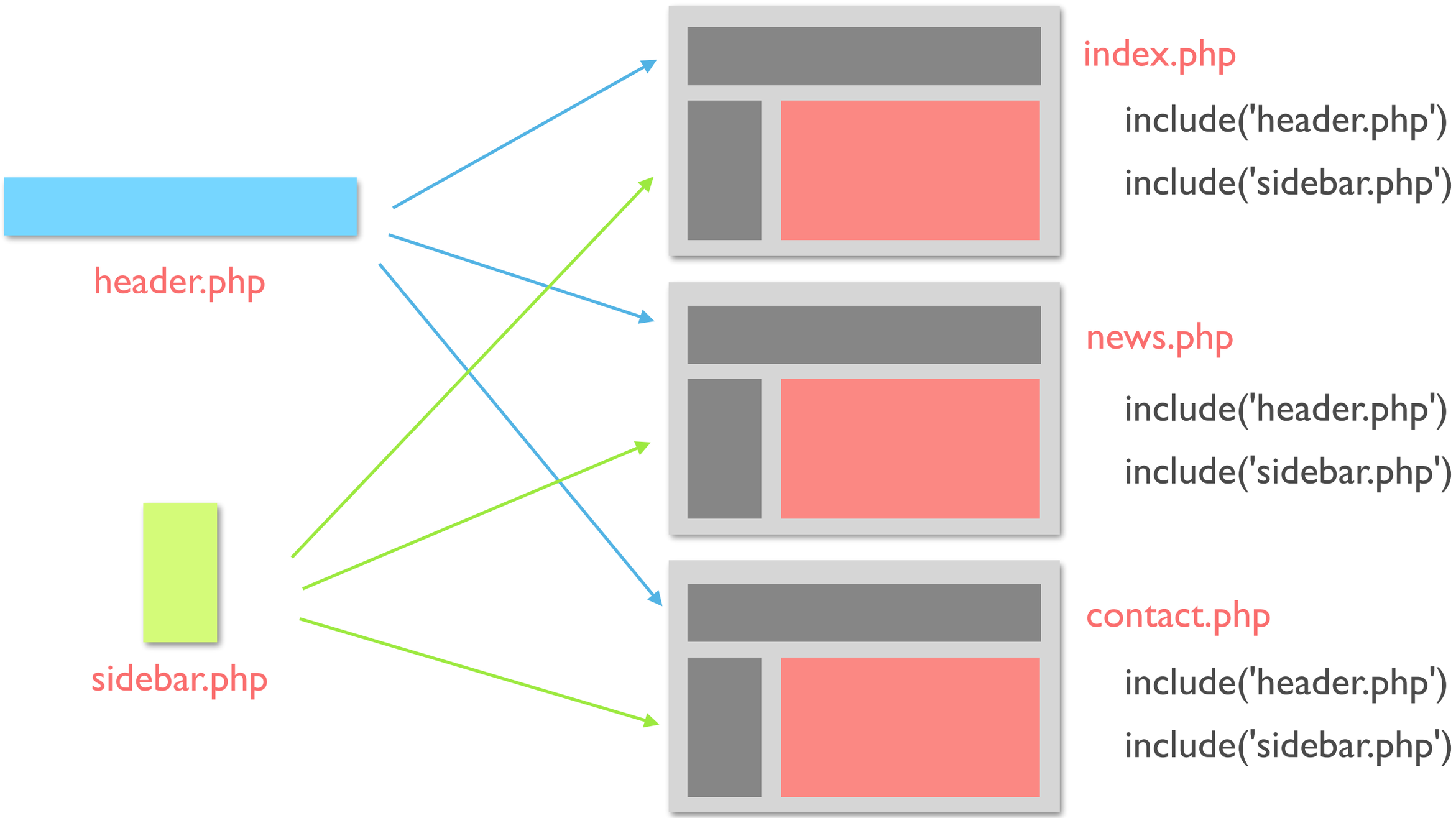
★ 參考文件：http://laravel.com/docs/5.1/blade

為什麼需要樣板？

- 在實作頁面時，很多頁面的 HTML 其實是重複的，為減少維護多份重複的原始碼，並讓開發更簡便，我們希望將頁面相同的區塊獨立出來，重覆使用
- 其實 PHP 本身就是一個樣板語言，內建就有 include 語法，可以用這個方式來將其他頁面檔案嵌入至當前畫面內

```
<!-- index.php -->
<body>
    <?php include("inc/header.php"); ?>
    <div class="container">
        <!-- 略 -->
    </div>
    <?php include("inc/footer.php"); ?>
</body>
```

include 概念圖


header.php
sidebar.php
index.php
include('header.php')
include('sidebar.php')
news.php
include('header.php')
include('sidebar.php')
contact.php
include('header.php')
include('sidebar.php')

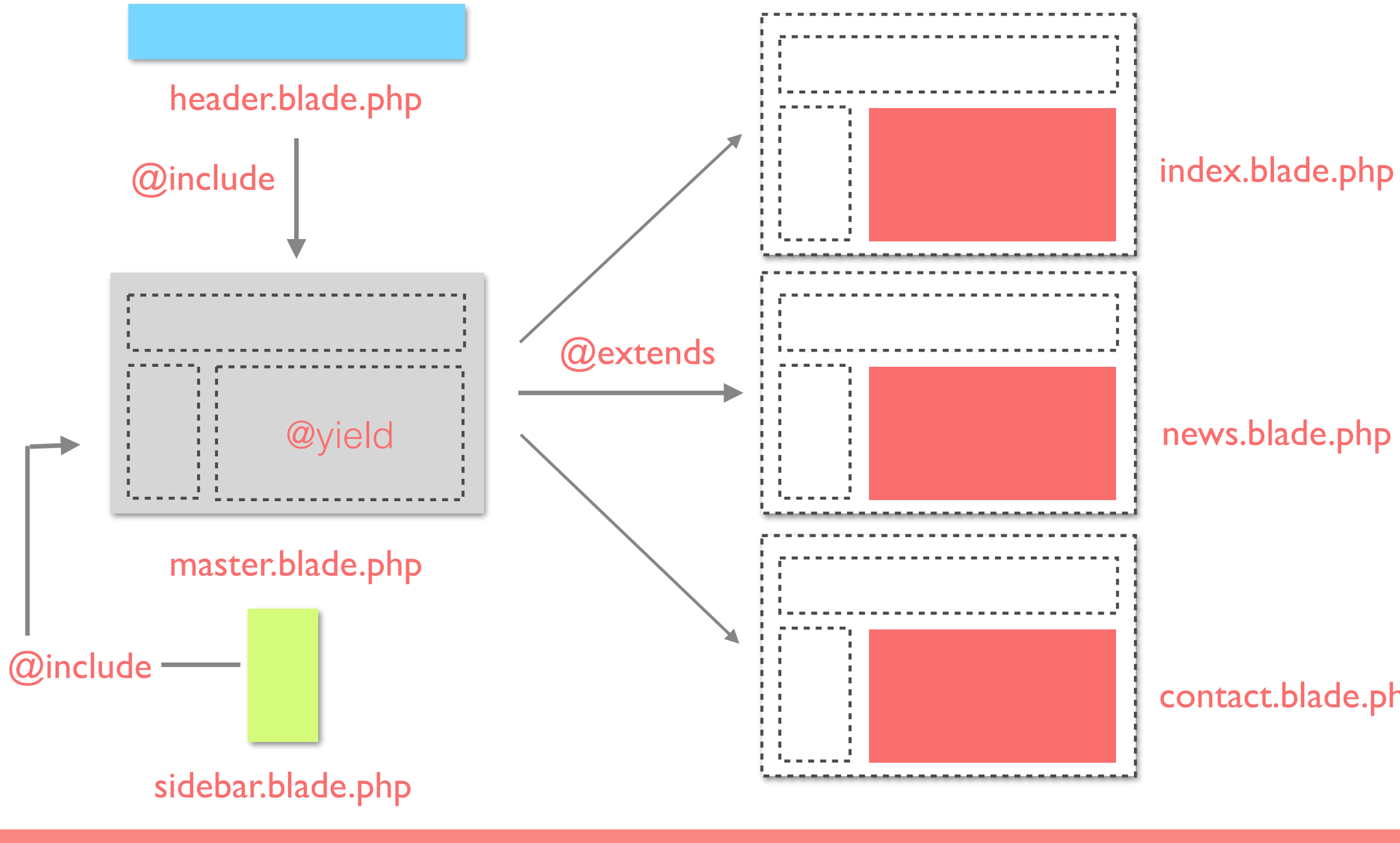
include 的極限

- include 只是單純的把指定的檔案內容如同複製貼上的方式放到當前的檔案內，本身並沒有容器的概念。換句話說，還是會有部份的內容是無法完美共用的
- include 也無法將動態資料傳遞進去，只能把檔案的內容單純的執行出來
- 因此，單純的 include 對開發來說是不夠用的，我們需要更好用的樣板引擎來控制頁面的顯示

Laravel 的樣板引擎

- Laravel 的樣板引擎名為 Blade
- 透過 Blade 語法，可以讓頁面有親屬關係，讓樣板繼承 (template inheritance) 變得簡單，且可將頁面重複的部份拆分成區塊 (section)，再引入至頁面對應的位置即可
- 可以傳遞參數到任一 Blade 樣板裡，在 Blade 樣板內也可以執行所有 PHP 內建的函數及 Laravel 的語法

使用 Blade Layout 後



怎麼用 Blade？

- 使用 Blade 非常的容易，只要將原本放在 resources/views 底下的 *.php 的檔名改成 *.blade.php。在畫面輸出時，Laravel 自動就會使用 Blade 語法分析器來處理這些檔案的內容
- Blade 規劃流程：
  - 先定義主 Layout
  - 定義共用區塊的內容
  - 再定義子 Layout 裡使用主 Layout 做延伸

定義主 Layout

- 在 resources/views 建立一個 layouts/master.blade.php
- 主 Layout 裡定義全站樣板的結構，以及要讓子 Layout 繼承填充的區塊位置

```
{{-- resources/views/layouts/master.blade.php --}}
<html>
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @include('layouts.partials.navigation')

        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

定義共用區塊

- 每個頁面都會使用到的共用區塊，可以再單獨拆分出來成單獨的 Blade 樣板
- 在 resources/views/layouts 底下新增 partials，裡面新增 navigation.blade.php 檔，並填入導覽列內容

```
{{-- resources/views/layouts/partials/navigation.blade.php --}}
<ul>
    <li>
        <a href="#">Item 1</a>
    </li>
    <li>
        <a href="#">Item 2</a>
    </li>
</ul>
```

定義子 Layout

- 在 resources/views 裡新增 *.blade.php
- 子 Layout 要先宣告繼承的主 Layout，並將要填充的區塊內容覆寫進去

```
@extends('layouts.master')

@section('title', 'Page Title')

@section('content')
    <p>This is my body content.</p>
@endsection
```

Blade 語法

- @ 開頭
  - 樣板控制 @yield、@extends…
  - 程式邏輯控制 @if、@foreach…
- { 開頭
  - 輸出內容 {{、{!!
  - 註解 {{--

Blade 樣板語法

- @yield('{section name}')
- @include('{view name}')
- @extends('{view name}')
- @section('{yeild name}', '{ string }')
- @section('{yeild name}')
- @endsction

Blade 邏輯控制

- 邏輯控制
  - `@if`、`@elseif`、`@else`、`@endif`
- 迴圈
  - `@for($i = 0; $i < 10; $++)`、`@endfor`
  - `@foreach($posts as $post)`、`@endforeach`
  - `@forelse($posts as $post)`、`@empty`、`@endforelse`

輸出資料

- `{{ $string }}`
  - 輸出前會做跳脫，安全預設
  - `{{ $name or 'Default' }}`
- `{!! $html !!}`
  - 輸出不會做跳脫，可用於印出 HTML
- 萬一真的要顯示 {{ 的時候怎麼辦？
  - 用 `@{{ raw data }}` 輸出

Blade 命名慣例

- 慣例僅供參考，Laravel 沒有強制規定要怎麼使用，只要符合 Laravel 原生設定就可以了
  - 主樣板放在 layouts 資料夾內
  - 區塊樣板放在 partials 資料夾內
  - 子樣板放在 {resources} 資料夾內

傳遞資料至 View 顯示

- 把變數傳給 View，讓 View 顯示動態資料

```php
// app/Http/routes.php
Route::get('/', function()
{
    $data = ['name' => 'Simon'];
    return view('home.index', $data);
});


// resources/views/home/index.blade.php
<!doctype html>
<html>
    <head><!-- 略 --></head>
    <body>
        <h1>My name is {{ $name }}</h1>
    </body>
</html>
```

様板製作/抽離

調整 views 結構/命名

- 新增 layouts、partials 資料夾
- 在 layouts 裡放入 master.blade.php
- 在 layouts/partials 裡放入 {partial}.blade.php
- 新增需要的各 {resource} 資料夾
- 把對應的 view 重新命名為 *.blade.php

設定主樣板

- 先將樣板內的總體 HTML 結構設定為主樣板

```
<!-- resources/views/layouts/master.blade.php -->
<!DOCTYPE html>
<html>
    <head>
        {{-- 略  --}}
        <title>@yield('title') | Blog</title>
    </head>
    <body>
        {{-- 略  —}}
        @include('layouts.partials.navigation')

        @yield('content')
        {{-- 略  --}}
    </body>
</html>
```

拔出重複的區塊

- 重複共用的區塊拔出來放到 partials 裡

```
<!-- resources/views/layouts/partials/navigation.blade.php -->
<div class="navigation">
    {{-- 略 --}}
</div>
```

設定子樣板

- 子樣板繼承主樣板，並填上/覆寫需要增加的區塊
- 可以使用 PHP 內建的函式來產生重覆的資料

```
<!-- resources/views/home/index.blade.php -->
@extends('layouts.master')

@section('title', '首頁')

@section('content')
    <h1>Home Page</h1>

    @foreach(range(1, 5) as $number)
    <h2>文章 - {{ $number }}</h2>
    @endforeach
@stop
```

修改 Route 路徑

- 因為調整過 views 資料夾內的檔名及資料夾結構，所以部份的 Route 可能也需要調整
- 全部完成後，請在瀏覽器裡驗證所有的修改都是正確無誤的

```
// app/Http/routes.php
Route::get('/', function()
{
    return view('home.index');
});
```

route('{name}', {prama})

- Laravel 提供的 helper function 之一
- 在其中指定 route 的名稱，就會自動回傳 URL
- 範例：

```
<a href="{{ route('home.index') }}">回首頁</a>
```

asset('{path}')

- Laravel 提供的 helper function 之一
- 在其中輸入素材的位置，就會自動回傳完整的 URL
- 範例：

```
<img src="{{ asset('img/logo.png') }}">
```

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？
  - MVC 基本概念及 Laravel 如何處理 View
  - Laravel 的 Blade 樣板語法
  - 整合專案樣板至 Laravel 內

Q & A

歡迎提問討論