

好孩子的 *php* 撰碼指南

范聖佑 (Shengyou Fan)
PHPCon China 2018
2018/05/19

范聖佑 shengyoufan@gmail.com

- 得寬科技 研究員
- Laravel 道場 主辦人
- JetBrains 技術專家
- 微軟最有價值專家
- 主要專長為 PHP / Laravel 後端技術



Laravel 道場



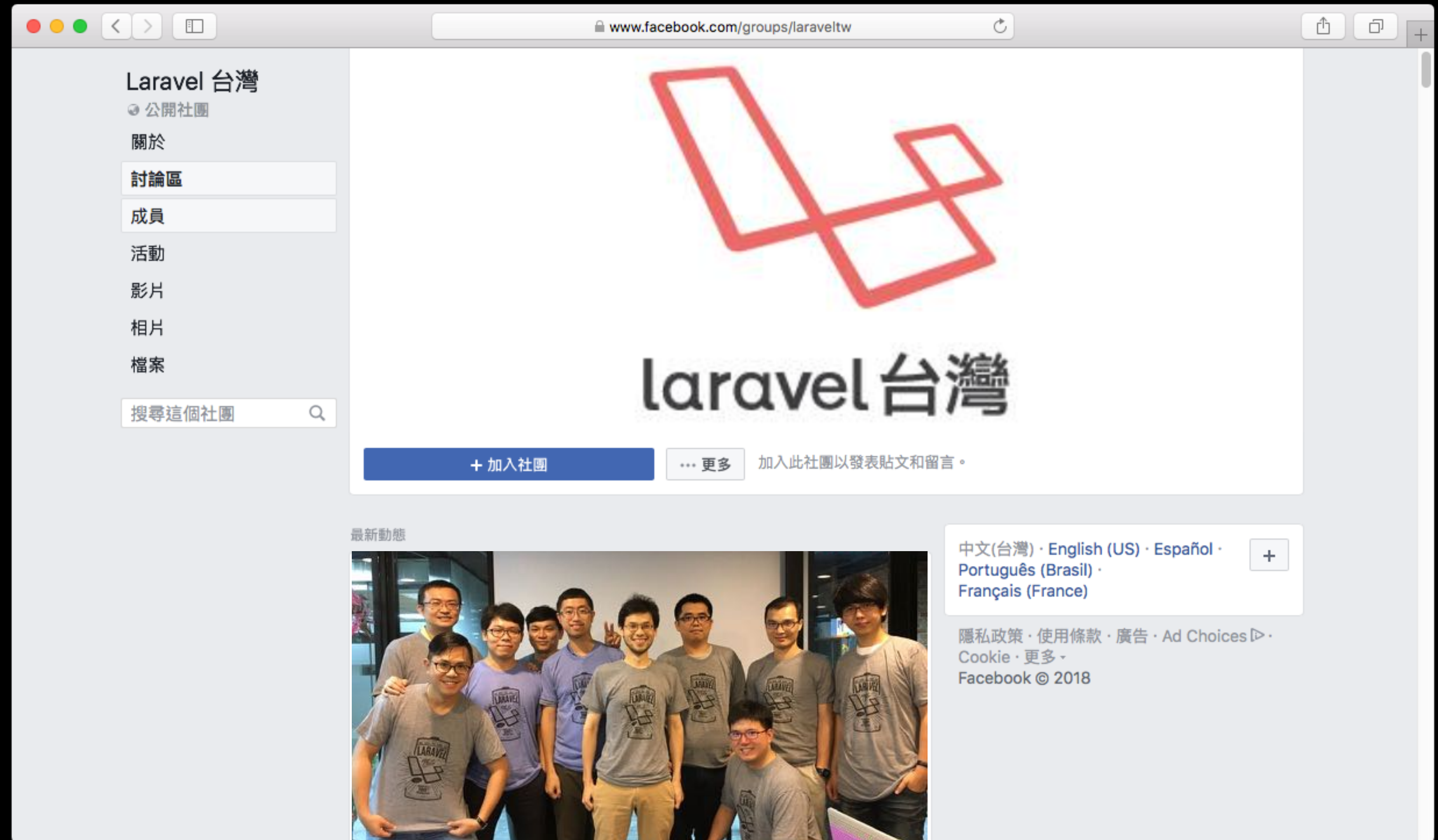
於 2014 年成立，致力於提供最專業的 Laravel 培訓及顧問服務。截至目前為止，已協助近 30 間學校及企業內部進行教學課程及技術導入。

<https://www.laravel-dojo.com/>



Laravel 台灣

成立於 2013 年，現有 6900 位用戶加入，每月不定期有 3 個社群活動，包括：PHP 也有 Day、Laradiner、Laradebut。



LaravelConf Taiwan

於 2017 年創辦第一屆，今年 7/8 (日) 舉辦第二屆，預計會有 450 位會眾參與。今年大會主題為「從想法到實現」，主題涵蓋套件生態系、套件開發技巧、測試、部署、效能、資安等主題。



首頁 / 關於 / 相簿 / 議程 / 講者 / 工作坊 / 交流趴 / 贊助商 / 場地

Laravel Conf

{{-- Taiwan 2018 --}}

from {{idea}} to {{production}}

2018/07/08(星期日) / 張榮發文教基金會 國際會議廳

50:22:24:42

Days Hours Minutes Seconds

手刀搶票去

主題大綱

一

從以下幾個不同的切入點，協助開發者
高效產出有品質的程式碼：

- 提升開發效率
- 統一風格規範
- 評估開發品質
- 測試開發成果
- 步驟自動完成

提升開發效率

覺得寫程式碼好花時間？試試這些工具：

- Boilerplate
- Scaffolding
- Code snippet

新專案初始化

我們是怎麼初始化一個 {框架} 專案的？

```
$ composer create-project \
{vendor/package}
```

```
1. php -n /usr/local/bin/composer create-project laravel/laravel --prefer-dist (rm)
~/Demo composer create-project laravel/laravel --prefer-dist
Installing laravel/laravel (v5.6.21)
- Installing laravel/laravel (v5.6.21): Loading from cache
Created project in /Users/shengyou/Demo/laravel
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 70 installs, 0 updates, 0 removals
- Installing vlucas/phpdotenv (v2.4.0): Loading from cache
- Installing symfony/css-selector (v4.0.9): Loading from cache
- Installing tijsverkoyen/css-to-inline-styles (2.2.1): Loading from cache
- Installing symfony/polyfill-php72 (v1.8.0): Loading from cache
- Installing symfony/polyfill-mbstring (v1.8.0): Loading from cache
- Installing symfony/var-dumper (v4.0.9): Loading from cache
- Installing symfony/routing (v4.0.9): Loading from cache
- Installing symfony/process (v4.0.9): Loading from cache
- Installing psr/log (1.0.2): Loading from cache
- Installing symfony/debug (v4.0.9): Loading from cache
- Installing symfony/http-foundation (v4.0.9): Loading from cache
- Installing paragonie/random_compat (v2.0.12): Loading from cache
- Installing symfony/event-dispatcher (v4.0.9): Loading from cache
- Installing symfony/http-kernel (v4.0.9): Loading from cache
- Installing symfony/finder (v4.0.9): Loading from cache
- Installing symfony/console (v4.0.9): Loading from cache
- Installing doctrine/lexer (v1.0.1): Loading from cache
- Installing egulias/email-validator (2.1.4): Loading from cache
- Installing swiftmailer/swiftmailer (v6.0.2): Loading from cache
- Installing ramsey/uuid (3.7.3): Loading from cache
```



現場演示

初始化之後

初始化專案後會做的事：

- 產生必要設定檔

```
$ cp .env.example .env
```

- 執行指令

```
$ artisan key:generate  
$ artisan package:discover
```

- 安裝套件

```
$ composer require {vendor/package}
```

```
1. php -n /usr/local/bin/composer require recca0120/laravel-tracy (php)  
~/Demo/laravel composer require recca0120/laravel-tracy  
Using version ^1.8 for recca0120/laravel-tracy  
./composer.json has been updated  
Loading composer repositories with package information  
Updating dependencies (including require-dev)  
Package operations: 5 installs, 0 updates, 0 removals  
- Installing tracy/tracy (v2.4.14): Loading from cache  
- Installing webmozart/path-util (2.3.0): Loading from cache  
- Installing webmozart/glob (4.1.0): Loading from cache  
- Installing recca0120/terminal (v1.6.21): Loading from cache  
- Installing recca0120/laravel-tracy (v1.8.17): Loading from cache  
tracy/tracy suggests installing https://nette.org/donate (Please support Tracy via a donation)  
Writing lock file  
Generating optimized autoload files  
> Illuminate\Foundation\ComposerScripts::postAutoload  
> @php artisan package:discover  
Discovered Package: fideloper/proxy  
Discovered Package: laravel/tinker  
Discovered Package: nunomaduro/collision  
Discovered Package: recca0120/laravel-tracy  
Discovered Package: recca0120/terminal  
Package manifest generated successfully.  
█
```



現場演示

以套件當基底做啟始包

Composer 的世界觀：

- 所有的專案都視為獨立套件
- 只要告訴它套件位置就抓得到

```
# 設定自訂套件來源
# ~/.composer/config.json
{
    "repositories": [
        {
            "type": "....",
            "url": "...."
        }
    ]
}

# 以自訂套件做專案啟始包
$ composer create-project {vendor/package}
```

綁定 Composer 事件

四大事件類型：

- Command Events
- Installer Events
- Package Events
- Plugin Events

<https://getcomposer.org/doc/articles/scripts.md>

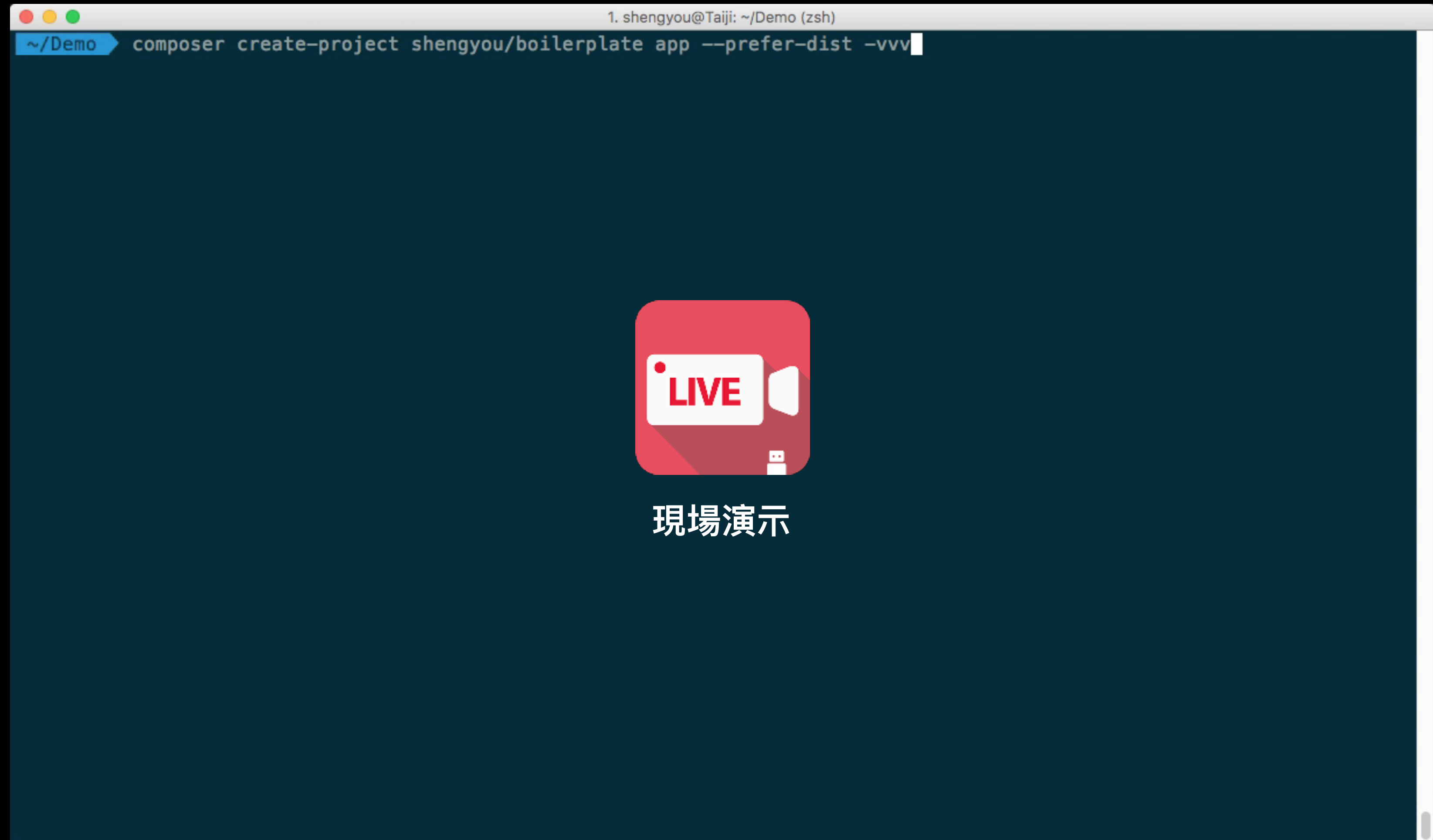
```
# 綁定事件
# ./composer.json
{
    "scripts": {
        "post-root-package-install": [
            "..."
        ],
        "post-create-project-cmd": [
            "..."
        ],
        "post-autoload-dump": [
            "..."
        ]
    }
}
```

```
# 強制觸發事件
$ composer run-script {event}
```

實作自己的啟始包

綜合以上機制，打造符合自己業務需求的專案啟始包：

- 建立儲存庫、提交、上 tag、Push 至遠端
- 定義自訂套件來源
- 用 Composer 建立專案
- 觸發 Script Events

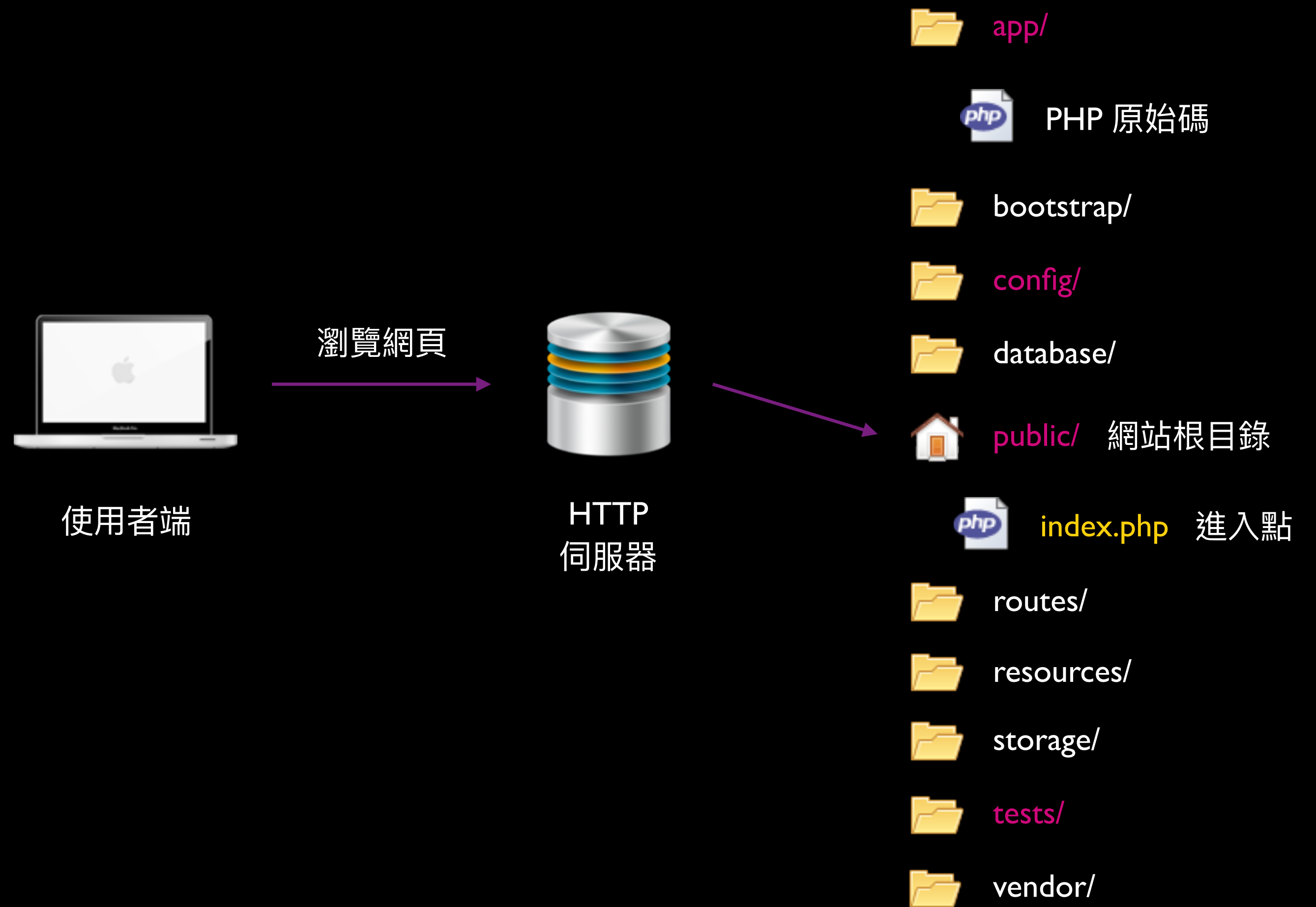


參考對象

可參考各框架的專案啟始包怎麼做？

- Laravel : `laravel/laravel`
- Symfony : `symfony/website-skeleton`
- Zend : `zendframework/skeleton-application`
- Yii : `yiisoft/yii2-app-basic`

Laravel 專案資料夾架構

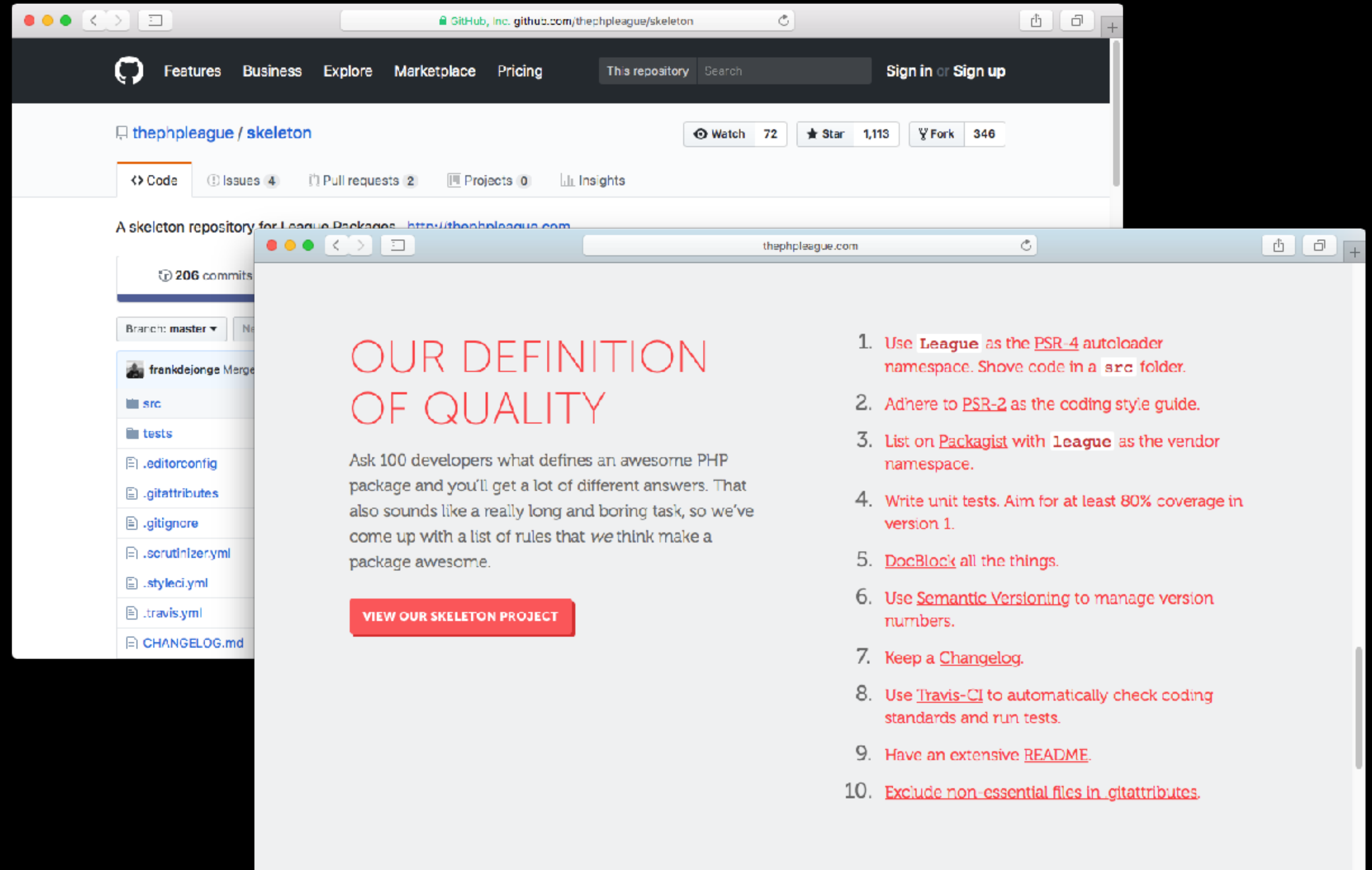


PHP League skeleton

可以參考 PHP League 所示範的專案架構：

<http://thephpleague.com>

<https://github.com/thephpleague/skeleton>

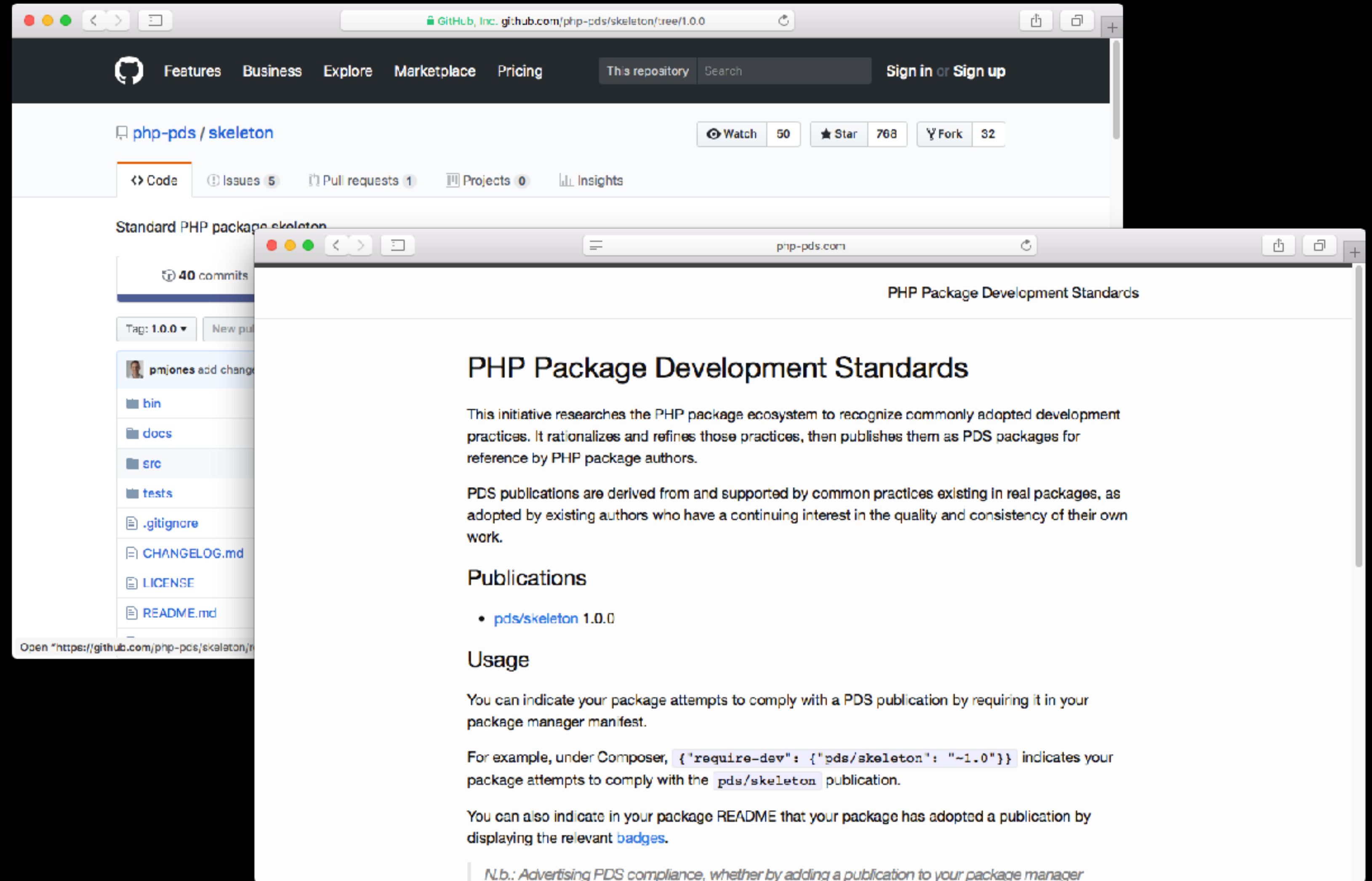


PHP Package development standard

可以參考 PHP Package development standard 來初始化 Package 專案：

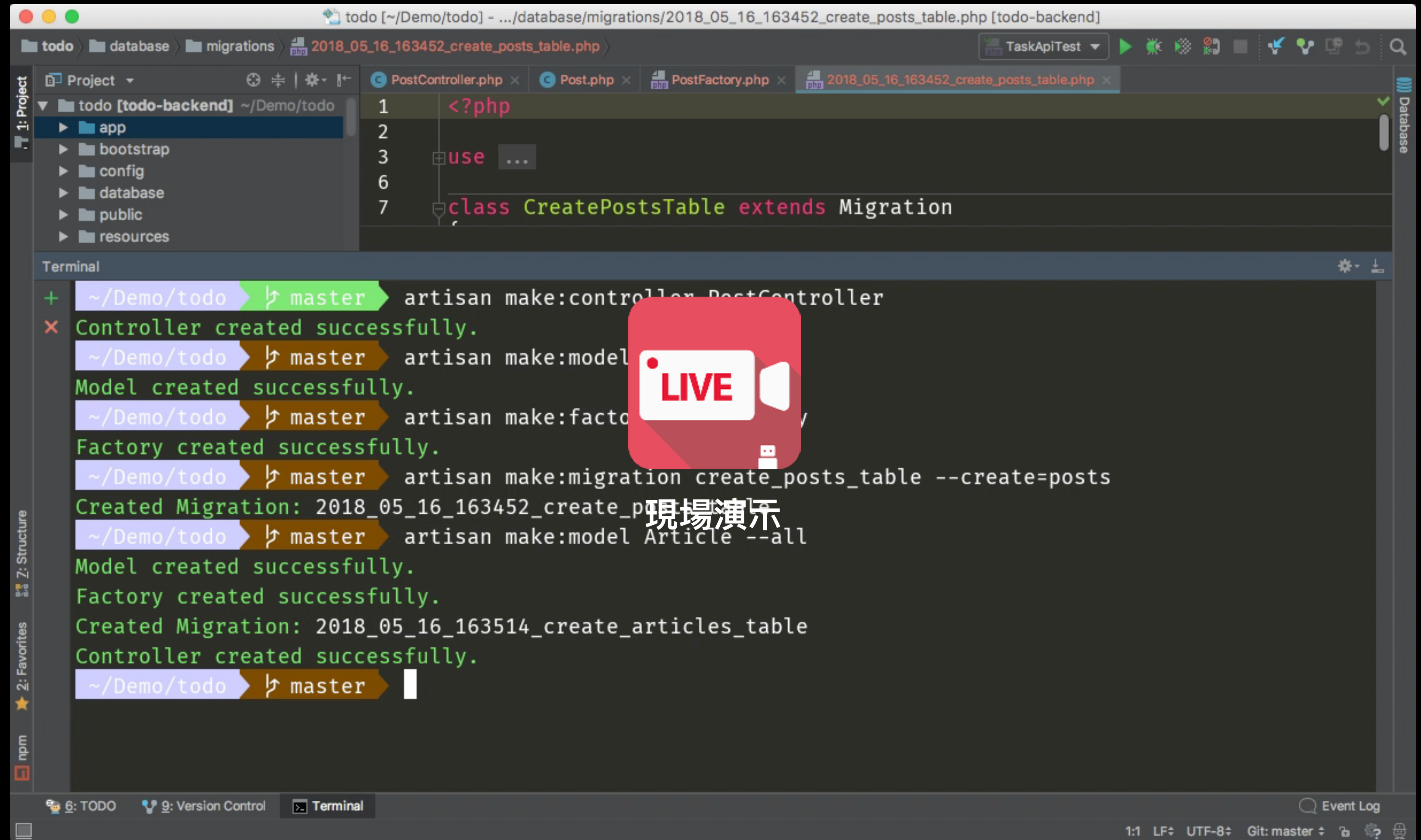
<http://php-pds.com/>

<https://github.com/php-pds>



快速產生必要檔案

在 Laravel 裡，有 artisan 指令可以快速產生實作所需的必要檔案。除了開檔、打字的時間外，也可以確保檔案放置在正確的路徑下。



The screenshot shows an IDE window with the following components:

- Project Explorer:** Shows the file structure of the 'todo' project, including 'app', 'bootstrap', 'config', 'database', 'public', and 'resources'.
- Code Editor:** Displays the file '2018_05_16_163452_create_posts_table.php' with the following code:

```
1 <?php
2
3 use ...
6
7 class CreatePostsTable extends Migration
```
- Terminal:** Shows the execution of several artisan commands and their successful outputs:

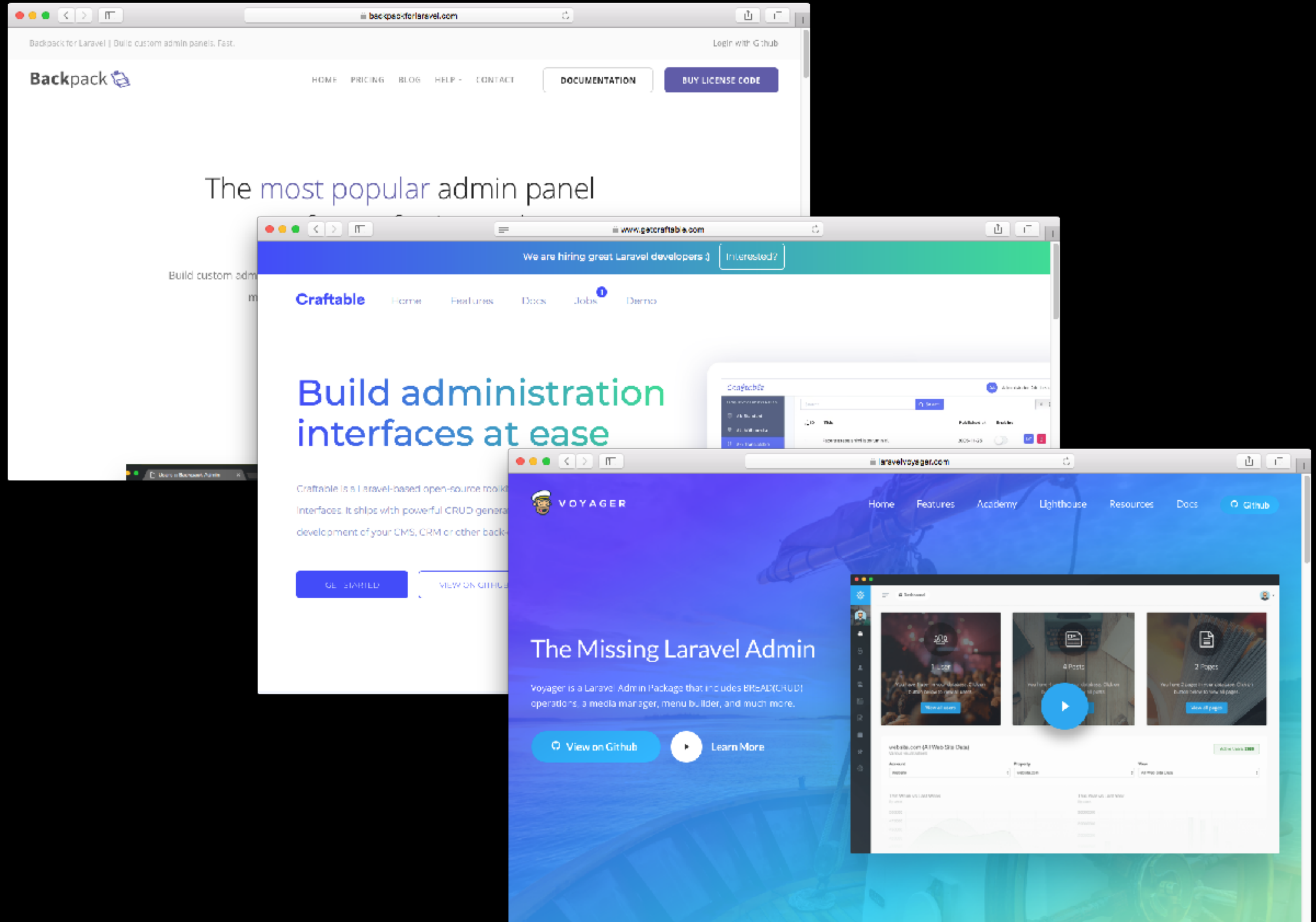
```
+ ~/Demo/todo master artisan make:controller PostController
x Controller created successfully.
~/Demo/todo master artisan make:model Article --all
Model created successfully.
~/Demo/todo master artisan make:factory PostFactory
Factory created successfully.
~/Demo/todo master artisan make:migration create_posts_table --create=posts
Created Migration: 2018_05_16_163452_create_posts_table
~/Demo/todo master artisan make:model Article --all
Model created successfully.
Factory created successfully.
Created Migration: 2018_05_16_163514_create_articles_table
Controller created successfully.
~/Demo/todo master
```

A red 'LIVE' watermark is visible in the center of the terminal output. The bottom status bar indicates the current file is '6: TODO', the version control system is '9: Version Control', and the terminal is active.

後台產生器

市面上有很多概念類似的後台產生器，可以快速滿足基本 CRUD 需求：

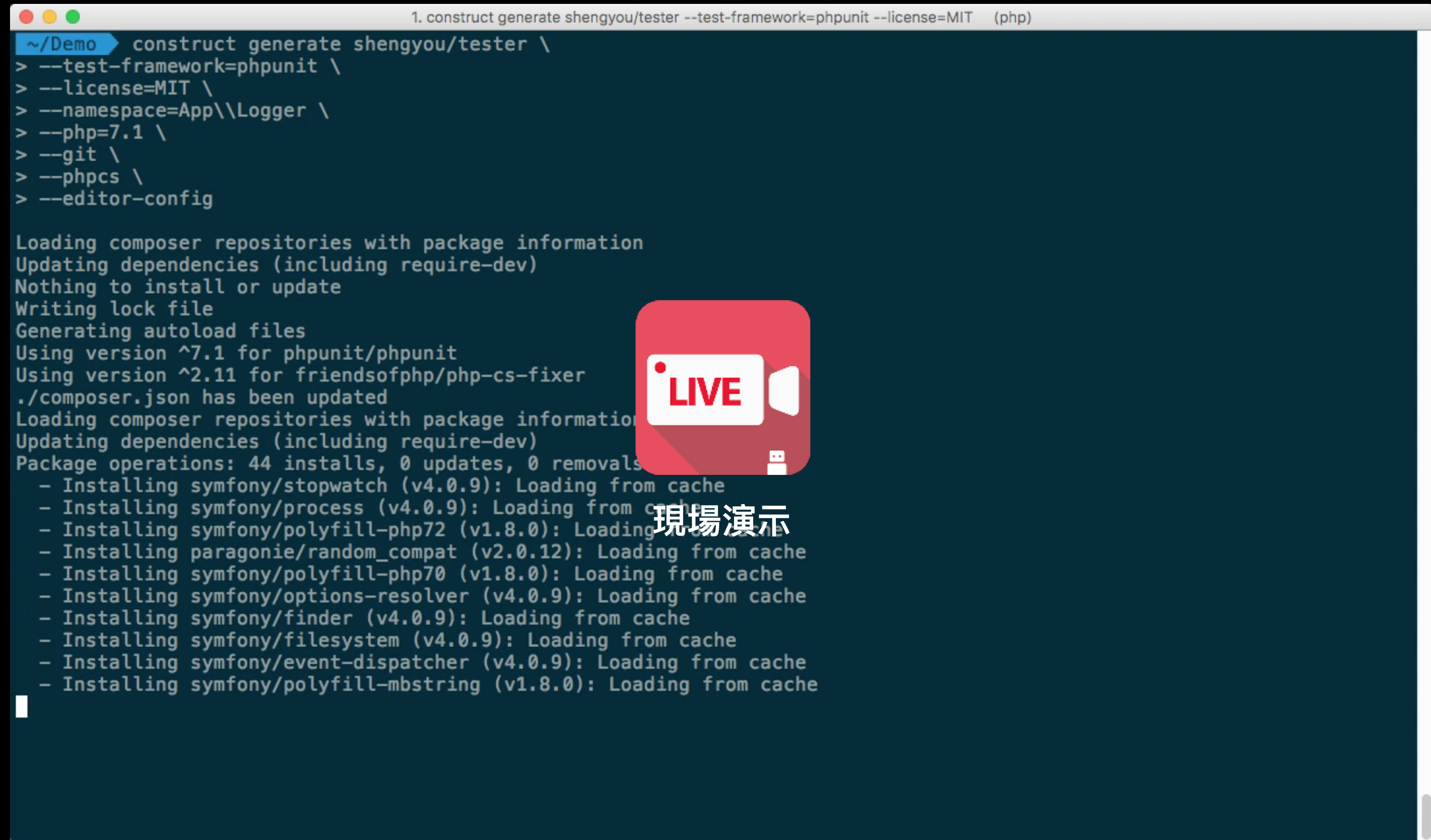
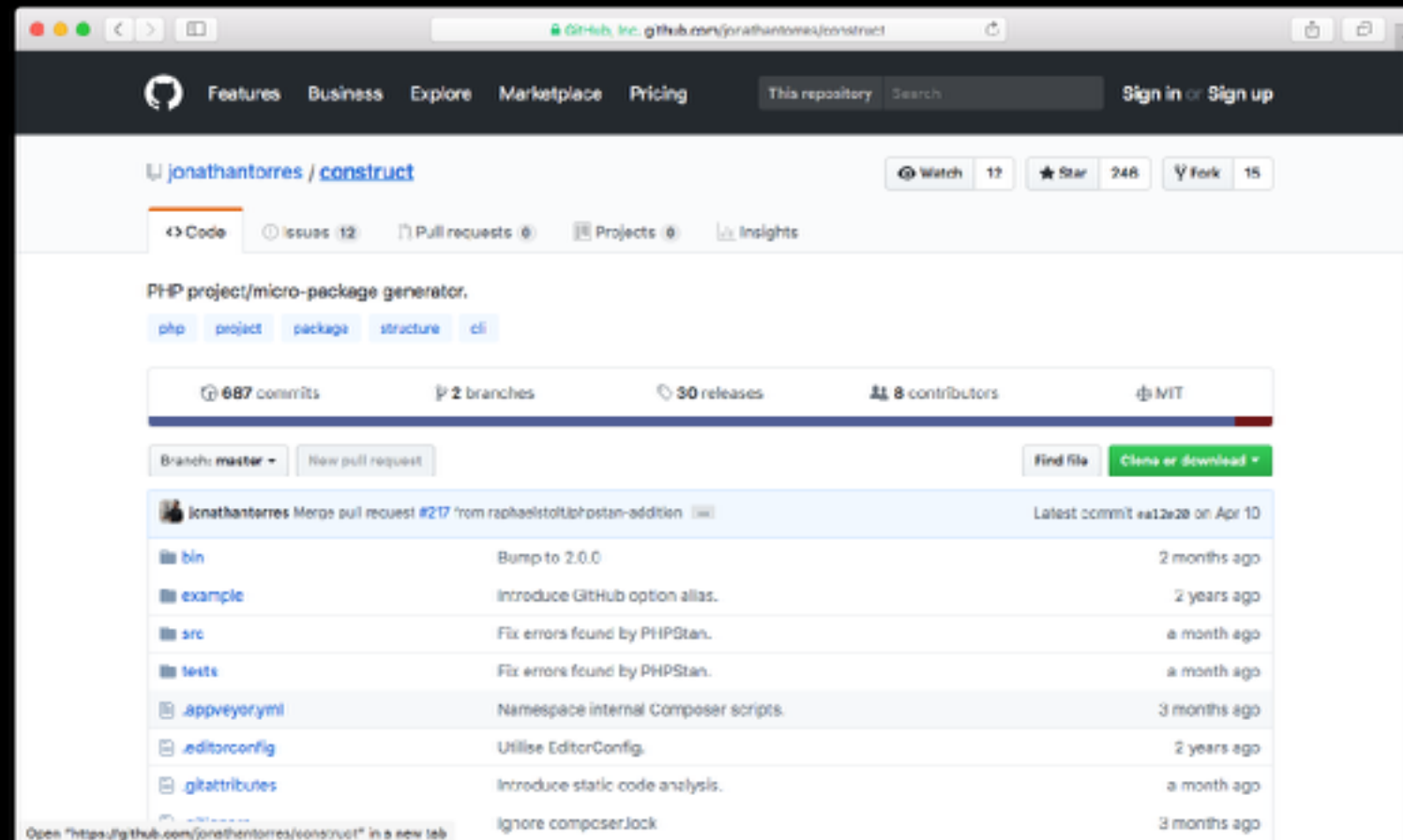
- <https://laravelvoyager.com/>
- <https://www.getcraftable.com/>
- <https://backpackforlaravel.com/>



實作自己的產生器

根據自己的實務需求，打造代碼產生器。
範例：

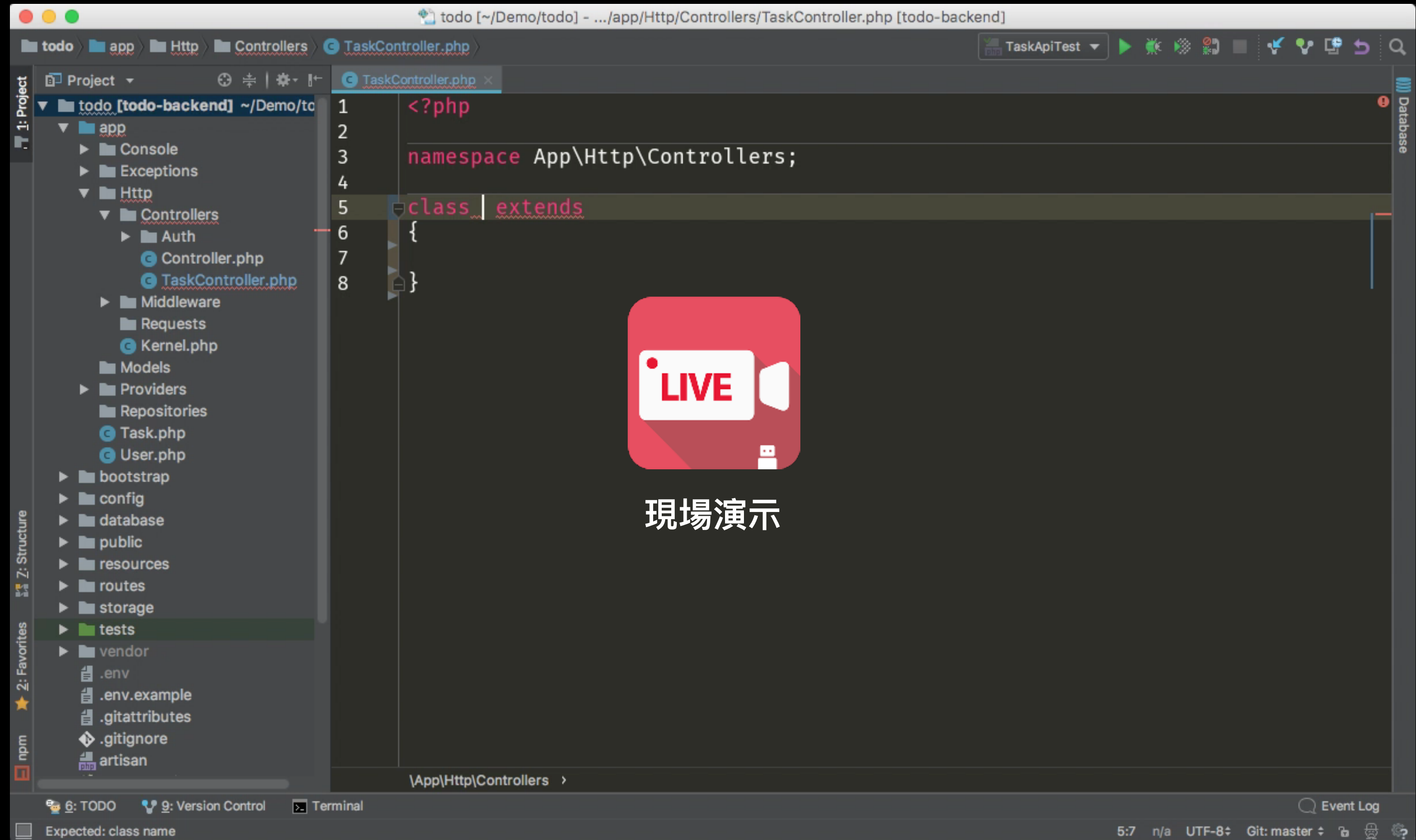
<https://github.com/jonathantorres/construct>



現場演示

快速產生代碼

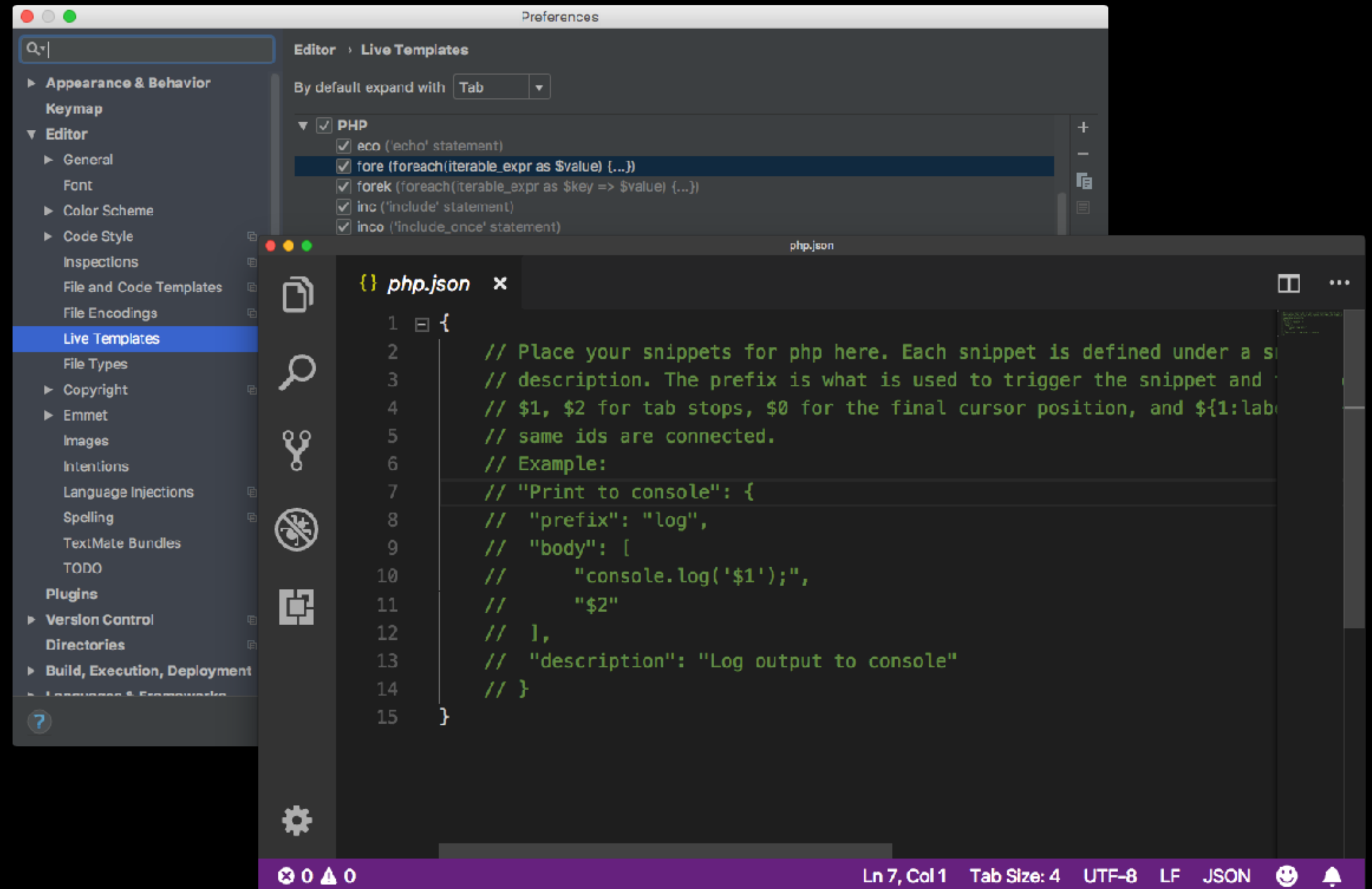
利用各 Editor/IDE 裡都有代碼區塊
(Code Snippet) 功能，快速產生常用程
式碼。



自訂程式碼區塊

在自己順手的 Editor/IDE 裡自訂程式碼區塊，增加產生程式碼的速度和減少打字時間。

花點時間設定、調校自己的工具，可以大大提升生產力。花更多時間在更有效益的事情上。



統一風格規範

別人寫的程式碼我都看不習慣，該怎麼辦？

- 建立撰碼風格規範
- 使用工具來修正現有程式碼

寫碼風格

—

程序員千古無解的討論：運作結果完全一樣的程式碼，喜歡哪一種寫碼風格？

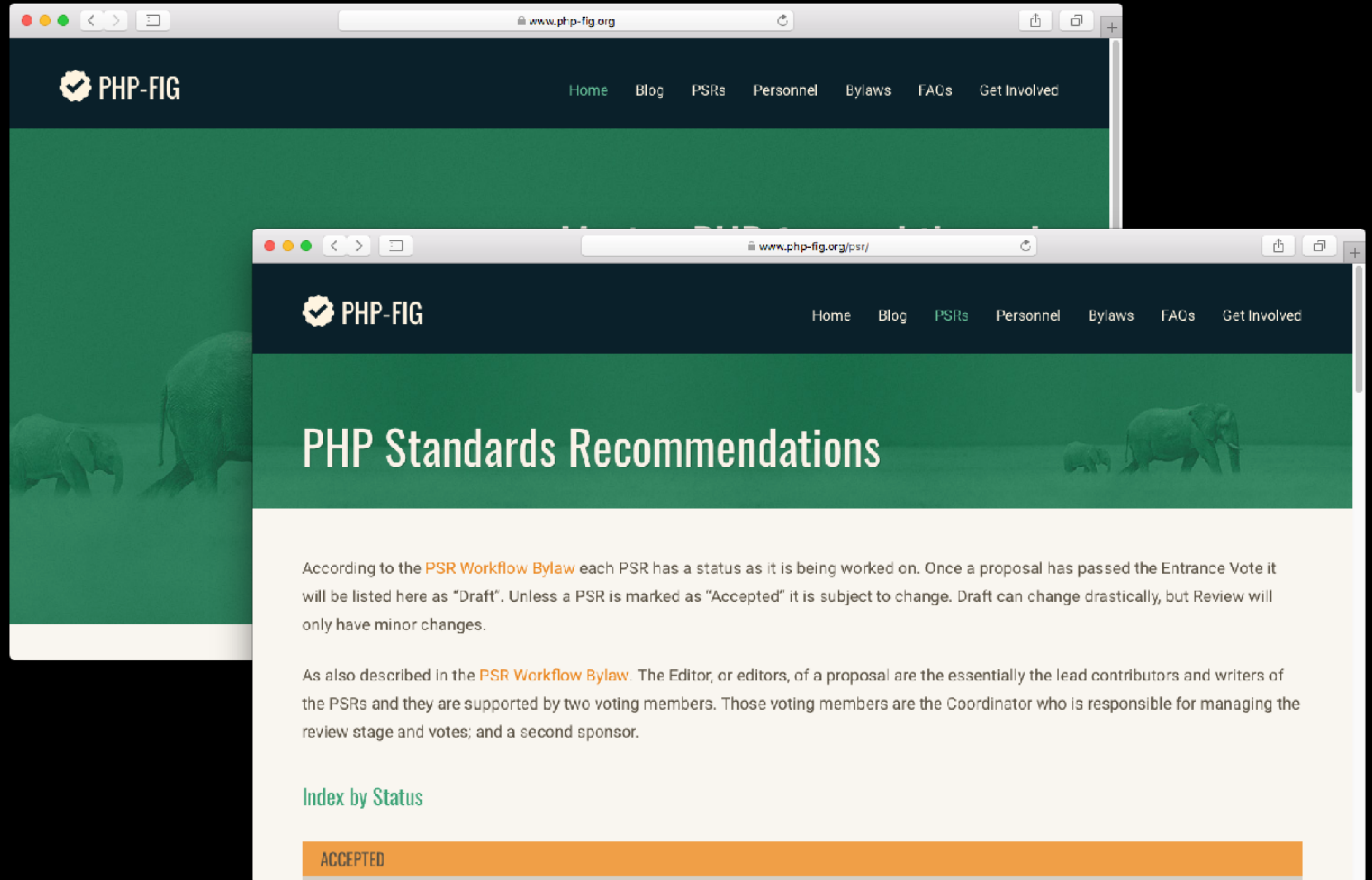
```
if ( ) {  
    // ...  
} elseif ( ) {  
    //...  
} else {  
    //...  
}
```

```
if()  
{  
    //...  
}  
else if()  
{  
    //...  
}  
else  
{  
    //...  
}
```

PHP-FIG

由 PHP-FIG 通過的 PHP Standards Recommendations 裡，其中 PSR-1 及 PSR-2 即針對撰碼風格做出規範。

<https://www.php-fig.org>

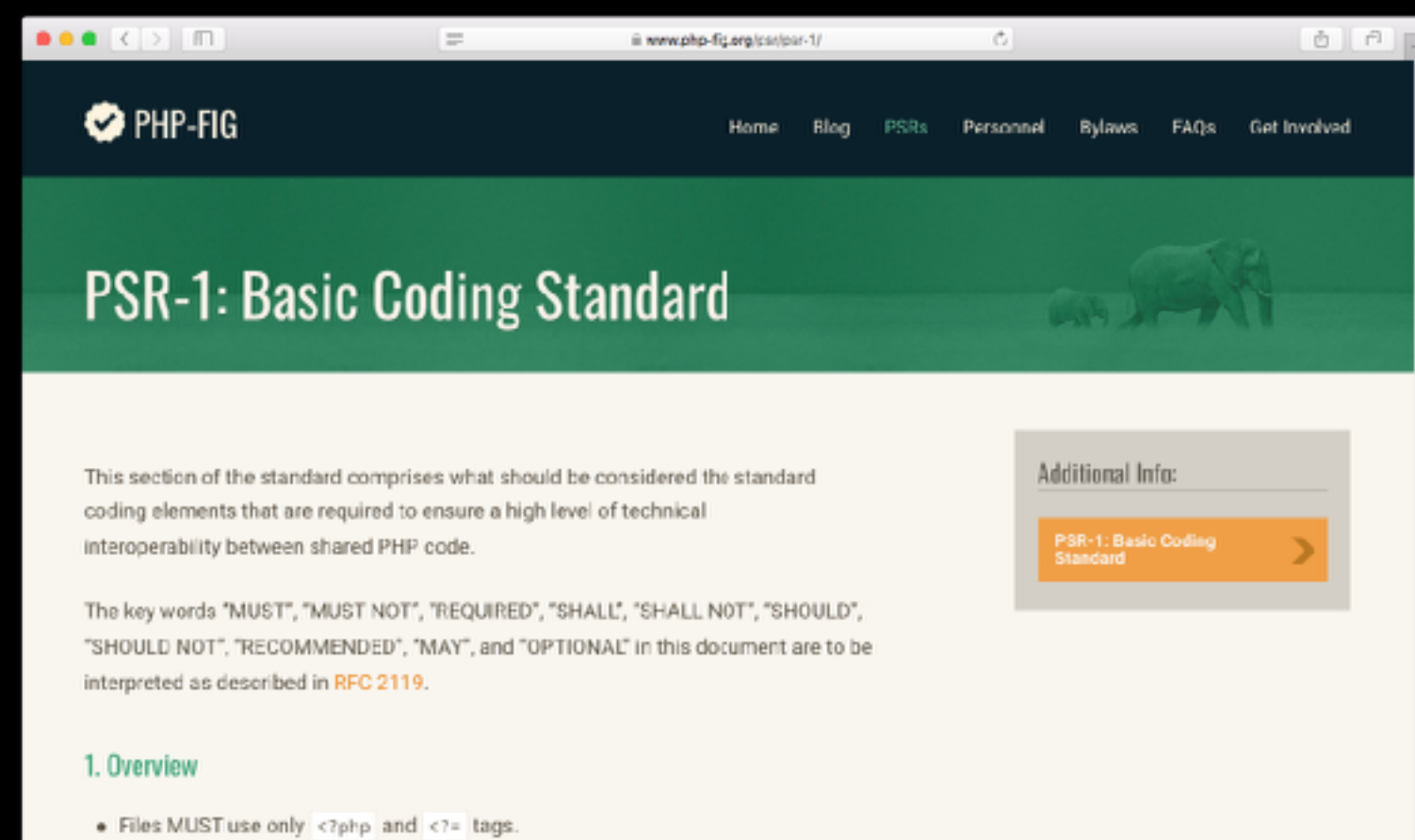


PSR-1

PSR-1: Basic Coding Standard

PSR-1：基礎撰碼規範

<https://www.php-fig.org/psr/psr-1/>



- ▶ 檔案裡一定只使用 `<?php` 和 `<?='` 標籤。
- ▶ 檔案一定在 PHP 程式碼裡使用沒有 BOM 的 UTF-8。
- ▶ 一個檔案裡只能在宣告 symbols（類別、函數、常數...等）或是產生 side-effects（如：產生輸出、變更 .ini 設定...等）中擇一，但不可兩者都做。
- ▶ 命名空間（Namespaces）和類別一定遵守 PSR-4 自動載入標準。
- ▶ 類別命名一定用首字大寫的駝峰式命名（StudlyCaps）。
- ▶ 類別常數命名一定用全大寫和底線組成。
- ▶ 類別方法命名一定用駝峰式命名（camelCase）。

範例：類別方法命名

類別方法一定以駝峰式命名 (camelCase)

```
# Foo.php
<?php

namespace Vendor\Model;

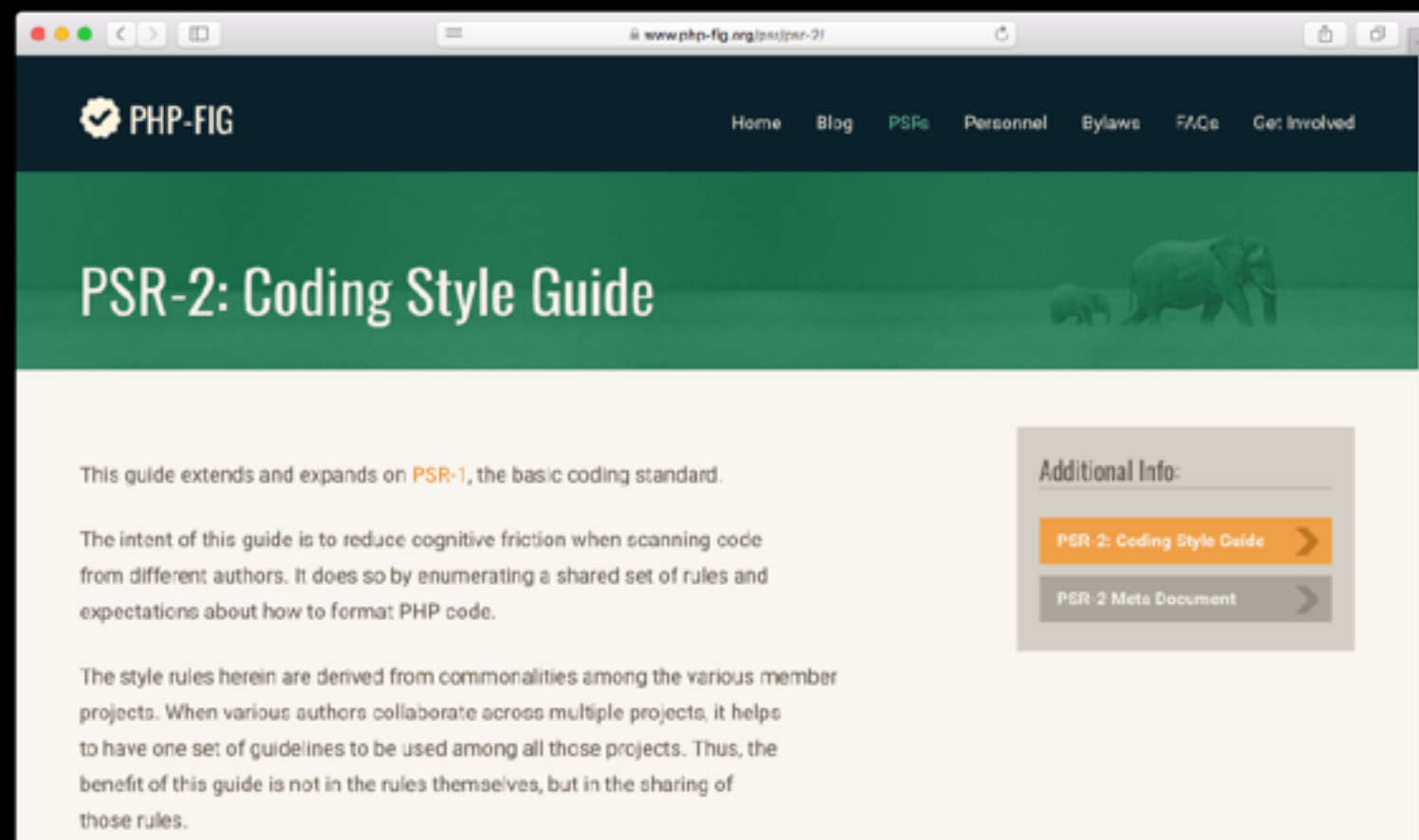
class Foo
{
    public function fooBar()
    {
        // method body
    }
}
```

PSR-2

PSR-2: Coding Style Guide

PSR-2：撰碼風格指南

<https://www.php-fig.org/psr/psr-2/>



- ▶ 程式碼一定遵守 PSR-1。
- ▶ 程式碼一定以 4 個空格，而不是用 tabs 當作縮排。
- ▶ 不可硬性規定單行長度；單行字元數一定限制在最多 120 字元左右；每行字元數一定在 80 字元或更少。
- ▶ 在 namespace 宣告後一定空一行，在全部 use 宣告完後一定空一行。
- ▶ 類別宣告的左大括號一定放在類別名稱的下一行，右大括號一定放在類別內容最後單獨一行。
- ▶ 方法宣告的左大括號一定放在方法名稱的下一行，右大括號一定放在方法內容最後單獨一行。
- ▶ 所有屬性和方法宣告一定加上可見性（Visibility）、abstract 和 final 一定寫在可見性之前、static 一定寫在可見性之後。
- ▶ 控制結構的關鍵字之後一定空一格；但方法和函數呼叫後不可空格。
- ▶ 控制結構的左大括號一定放在同一行，右大括號一定放在內容最後單獨一行。
- ▶ 控制結構的左小括號之後不可空格，右小括號之前也不可空格。

範例：PSR-2

else 和 elseif 要和上一個的右大括號放在同一行。為了讓所有控制結構關鍵字看起來是一個單字，一定使用關鍵字 elseif 而不是 else if。

for 及 foreach 陳述語法看起來結構如右，注意小括號、空格和大括號的位置。

```
<?php
```

```
# 控制結構
```

```
if ($expr1) {  
    // if body  
} elseif ($expr2) {  
    // elseif body  
} else {  
    // else body;  
}
```

```
# 迴圈
```

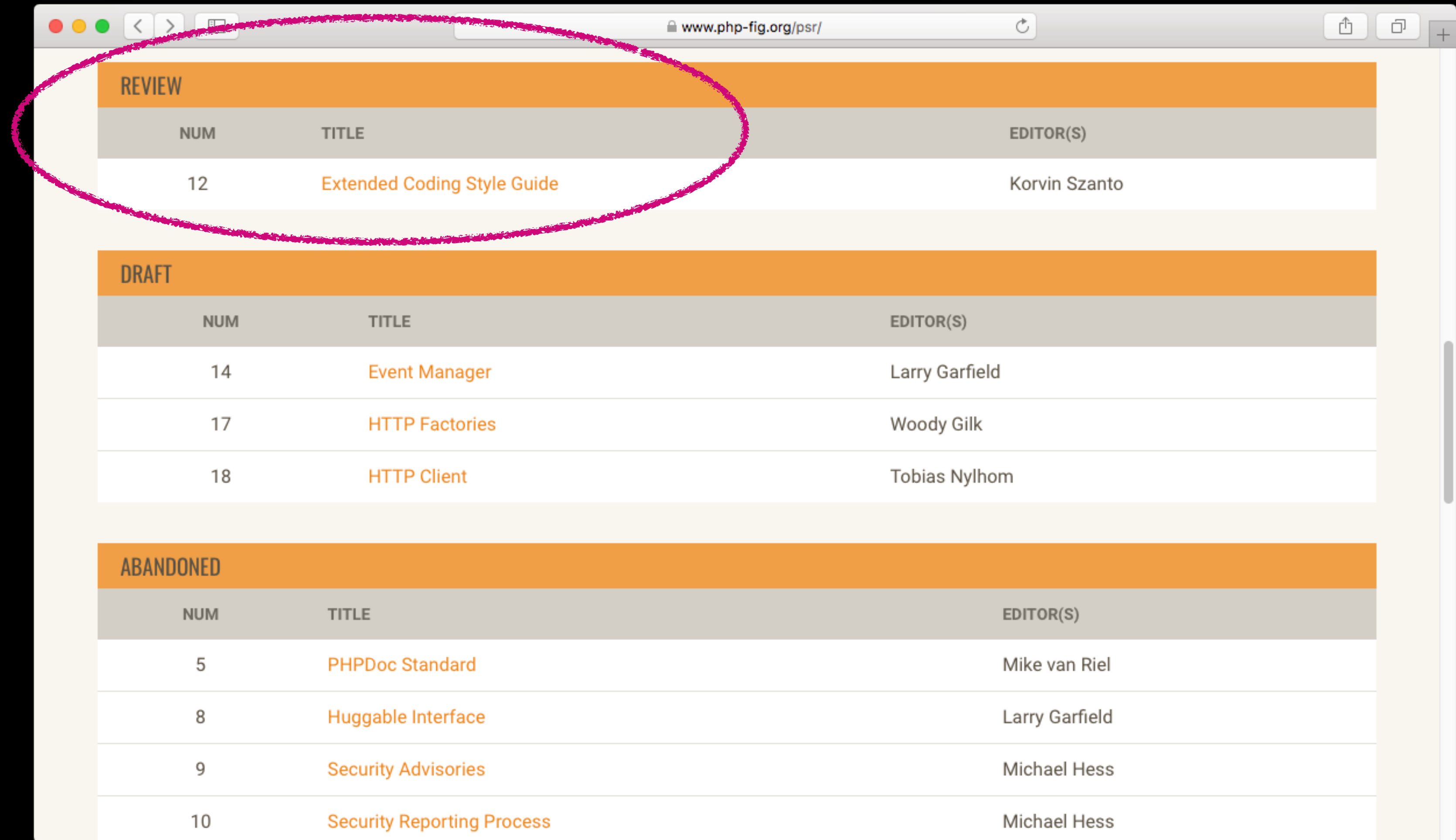
```
for ($i = 0; $i < 10; $i++) {  
    // for body  
}
```

```
foreach ($iterable as $key => $value) {  
    // foreach body  
}
```

PSR-12

PSR-12: Extended Coding Style Guide
(PSR-12：延伸撰碼風格指南)

針對 PHP 7 新語法所提出的撰碼風格規範，目前正在審核階段。



The screenshot shows the website www.php-fig.org/psr/ with three sections: REVIEW, DRAFT, and ABANDONED. The 'REVIEW' section is circled in red and contains one entry. The 'DRAFT' section contains three entries, and the 'ABANDONED' section contains four entries. Each entry is a row in a table with columns for NUM, TITLE, and EDITOR(S).

REVIEW		
NUM	TITLE	EDITOR(S)
12	Extended Coding Style Guide	Korvin Szanto

DRAFT		
NUM	TITLE	EDITOR(S)
14	Event Manager	Larry Garfield
17	HTTP Factories	Woody Gilk
18	HTTP Client	Tobias Nyholm

ABANDONED		
NUM	TITLE	EDITOR(S)
5	PHPDoc Standard	Mike van Riel
8	Huggable Interface	Larry Garfield
9	Security Advisories	Michael Hess
10	Security Reporting Process	Michael Hess

EditorConfig

幫助開發者在不同的 Editor/IDE 之間採用一致的撰碼風格。



範例：.editorconfig

語法規則：

- 使用 ini 語法格式
- 使用 UTF-8、LF 作為檔案及換行格式
- 在區段裡可使用 [及] 標記套用的路徑 (類似 .gitignore)
- 以斜線 (/) 標示路徑
- # 或 ; 為註解、單獨佔一行

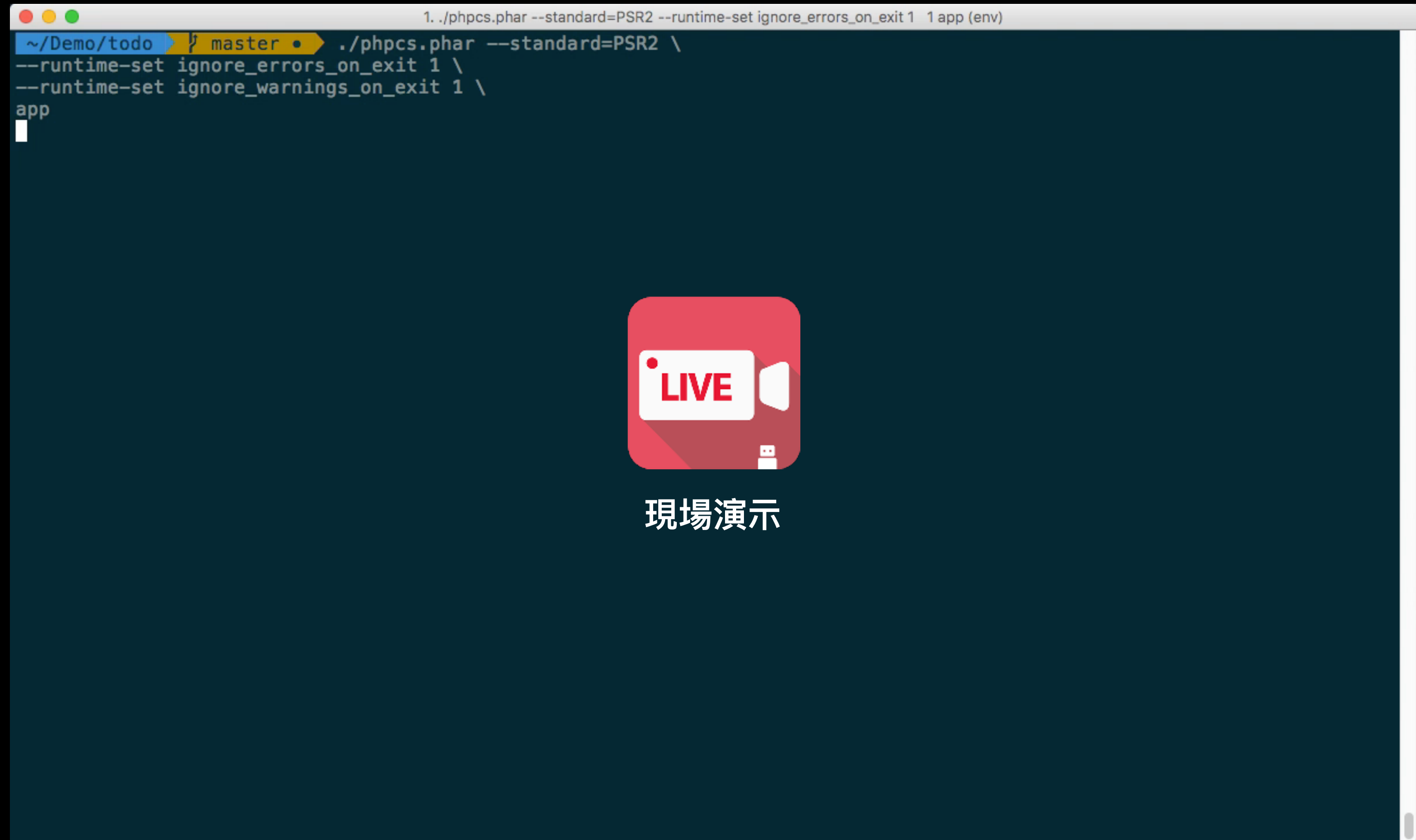
```
# This file is for unifying the coding style for  
different editors and IDEs  
# editorconfig.org
```

```
root = true
```

```
[*.php]  
charset = utf-8  
indent_size = 4  
indent_style = space  
end_of_line = lf  
insert_final_newline = true  
trim_trailing_whitespace = true
```

CodeSniffer

Code Sniffer 可以依照設定的語法規範，掃描程式碼後列出修正清單。也可以依照預設擴充語法規範，做為團隊撰碼時的檢查器。



撰碼風格修正器

手動修正撰碼風格耗時又費力，PHP 社群已有成熟的撰碼風格修正器，安裝後即可使用。

```
# 安裝 phpcbf
```

```
$ wget https://.../phpcbf.phar
```

```
$ chmod a+x phpcbf.phar
```

```
# phpcbf 自動修正
```

```
$ [php] bin/phpcbf.phar --standard=PSR2 \
--runtime-set ignore_errors_on_exit 1 \
--runtime-set ignore_warnings_on_exit 1 \
{src}
```

```
# 安裝 php-cs-fixer
```

```
$ wget http://.../php-cs-fixer-v2.phar
```

```
$ chmod a+x php-cs-fixer-v2.phar
```

```
# php-cs-fixer 自動修正
```

```
$ [php] php-cs-fixer.phar fix {src} \
--rules=@PSR2
```

使用工具修正撰碼風格

使用 phpcbf 或 php-cs-fixer 等工具來協助排版代碼，可以綁定在 Editor/IDE 存檔時觸發。

```
1. shengyou@Taiji: ~/Demo/laravel (vim)
~/Demo/laravel master ➤ vim app/Http/Controllers/TaskController.php
~/Demo/laravel master ➤ wget https://cs.sensiolabs.org/download/php-cs-fixer-v2.phar
--2018-05-13 17:52:56-- https://cs.sensiolabs.org/download/php-cs-fixer-v2.phar
Resolving cs.sensiolabs.org (cs.sensiolabs.org)... 185.199.110.153, 185.199.111.153, 185.199.108.153, ...
Connecting to cs.sensiolabs.org (cs.sensiolabs.org)|185.199.110.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1687357 (1.6M) [application/octet-stream]
Saving to: 'php-cs-fixer-v2.phar'

php-cs-fixer-v2.phar      100%[=====>]    1.61M  1.91MB/s   in 0.8s

2018-05-13 17:52:57 (1.91 MB/s) - 'php-cs-fixer-v2.phar' saved [1687357/1687357]

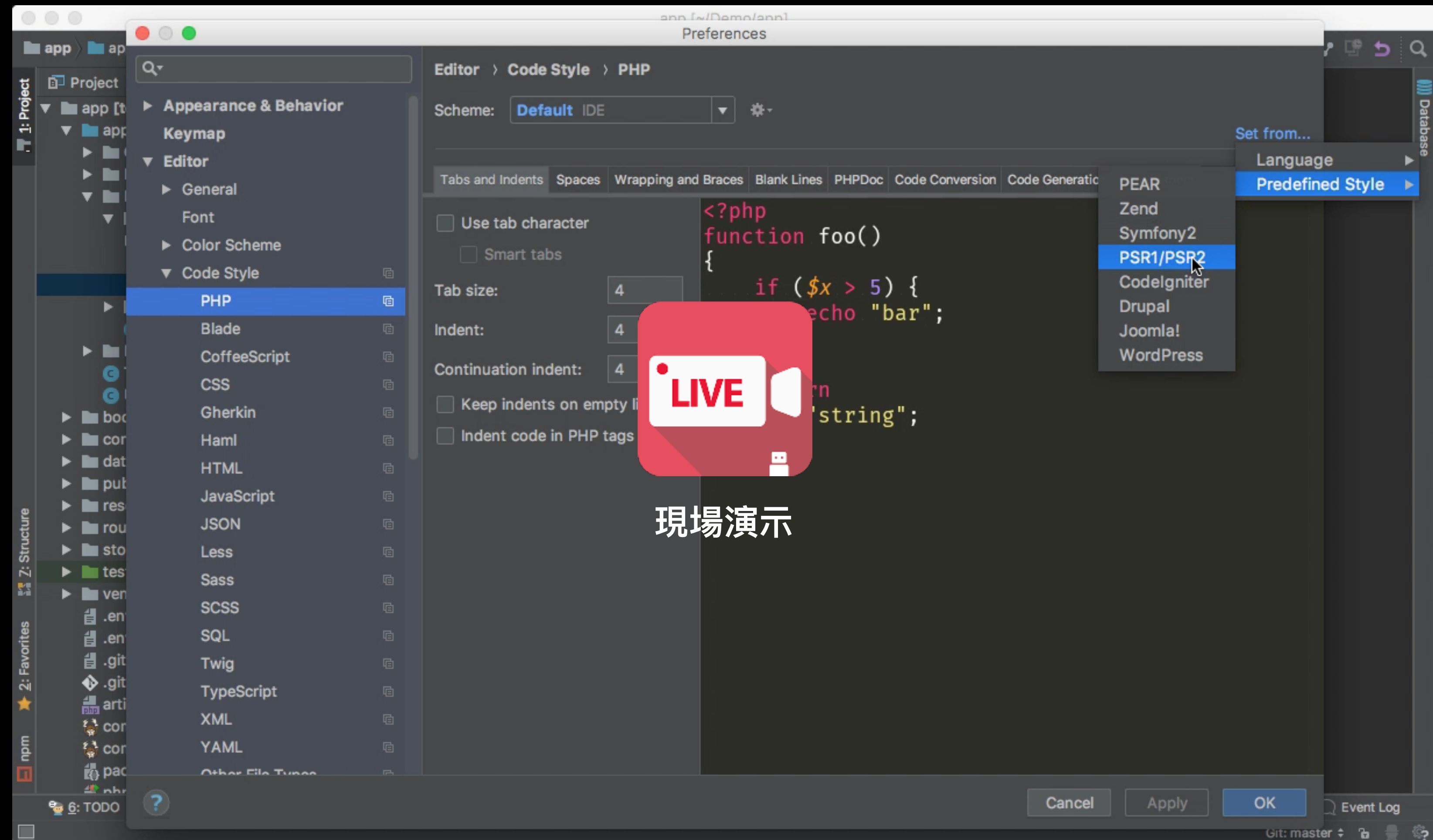
~/Demo/laravel master ➤ chmod a+x php-cs-fixer-v2.phar
~/Demo/laravel master ➤ php php-cs-fixer-v2.phar --rules=@PSR2
Loaded config default.
 1) app/Http/Controllers/TaskController.php

Fixed all files in 0.100 seconds, 8.000 MB memory used
~/Demo/laravel master ➤ vim app/Http/Controllers/TaskController.php
```

現場演示

Editor/IDE 的代碼排版、整理功能

大多數的 Editor/IDE 都有代碼排版、重新整理、排列等功能，只需一個按鍵就可以省下大量時間。



評估開發品質

代碼是寫完了，但...寫得好嗎？

- Code Quality
- Code Analysis

語法檢查

PHP 內建就有 Linter，大多數的 Editor/IDE 也都有對應的支援，可在寫程式碼的當下就避開錯誤。

<https://github.com/JakubOnderka/PHP-Parallel-Lint>

```
1. shengyou@Taiji: ~/Demo/app (zsh)
~/Demo/app master • parallel-lint --exclude vendor .
PHP 7.1.17 | 10 parallel jobs
X.X... 6/6 (100 %)

Checked 6 files in 0.1 seconds
Syntax error found in 2 files

-----
Parse error: ./config/local-example.php:8

    ]
  >
Unexpected end of file, expecting ';' in ./config/local-example.php on line 8
-----
Parse error: ./src/Controllers/HomeController.php:10

    {
      return true
    }
  >
}

Unexpected '}', expecting ';' in ./src/Controllers/HomeController.php on line 10
X ~/Demo/app master • vim config/local-example.php
~/Demo/app master • vim src/Controllers/HomeController.php
~/Demo/app master • parallel-lint --exclude vendor .
PHP 7.1.17 | 10 parallel jobs
..... 6/6 (100 %)

Checked 6 files in 0 seconds
No syntax error found
```



現場演示

Mess Detector

檢查程式碼內各種可能不良的實作提出警告提示。

<https://phpmd.org/>

```
1. shengyou@Taiji: ~/Demo/app (zsh)
~/Demo/app master • ./phpmd.phar src text cleancode, codesize, controversial, design, naming, unusedcode
/Users/shengyou/Demo/app/src/Controllers/HomeController.php:9  Avoid variables with short names like $a. Configured minimum length is 3.
/Users/shengyou/Demo/app/src/Controllers/HomeController.php:9  Avoid unused local variables such as '$a'.
/Users/shengyou/Demo/app/src/Controllers/HomeController.php:10  Avoid excessively long variable names like $longLongLongLongLongVar. Keep variable name length under 20.
/Users/shengyou/Demo/app/src/Controllers/HomeController.php:10  Avoid unused local variables such as '$longLongLongLongLongVar'.
x ~/Demo/app master •
```



現場演示

Copy/Paste Detector

偵測程式碼內有無重複 (剪下貼上) 的程式碼？

<https://github.com/sebastianbergmann/phpcpd>

```
1. shengyou@Taiji: ~/Demo/app (zsh)
~/Demo/app master ./phpcpd.phar . --exclude=vendor --progress
phpcpd 4.0.0 by Sebastian Bergmann.

7/7 [=====] 100%

Found 1 clones with 59 duplicated lines in 1 files:

- /Users/shengyou/Demo/app/Math.php:75-134
  /Users/shengyou/Demo/app/Math.php:139-198

23.23% duplicated lines out of 254 total lines of code.
Time: 37 ms, Memory: 4.00MB
x ~/Demo/app master
```



現場演示

PHPStan

透過靜態分析器，不需執行程式碼就可以做語法檢查。

<https://github.com/phpstan/phpstan>

```
1. shengyou@Taiji: ~/Demo/app (zsh)
~/Demo/app master vim src/Controllers/HomeController.php
~/Demo/app master composer require --dev phpstan/phpstan
Using version ^0.9.2 for phpstan/phpstan
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Nothing to install or update
Writing lock file
Generating autoload files
ocramius/package-versions: Generating version class...
ocramius/package-versions: ...done generating version class
~/Demo/app master phpstan analyse src --level=7
1/1 [████████████████████] 100%

-----
Line   src/Controllers/HomeController.php
-----
7      Parameter $task of method App\Controllers\HomeController::index() has invalid typehint type App\Controllers\Task.
9      Call to static method find() on an unknown class App\Controllers\Task.
-----

[ERROR] Found 2 errors

x ~/Demo/app master
```

現場演示

一次跑全部：phpqa

覺得太多工具要跑很麻煩？直接一個工具幫你跑全部！

<https://github.com/EdgedesignCZ/phpqa>

1. shengyou@Taiji: ~/Demo/app (git)

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/phploc" --exclude=vendor "src" --log-xml "build//phploc.xml"

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/phpcs" -p --standard=PSR2 --ignore=*/vendor/* "src" --extensions=php --report-checkstyle="build//checkstyle.xml"

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/phpmd" "src" xml "/Users/shengyou/.composer/vendor/edgedesign/phpqa/app/phpmd.xml" --exclude /vendor/ --suffixes php --reportfile "build//phpmd.xml"

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/pdepend" --jdepend-xml="build//pdepend-jdepend.xml" --summary-xml="build//pdepend-summary.xml" --dependency-xml="build//pdepend-dependencies.xml" --jdepend-chart="build//pdepend-jdepend.svg" --overview-pyramid="build//pdepend-pyramid.svg" --suffix=php --ignore=*/vendor/ "src"

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/phpcpd" --progress --exclude=vendor "src" --min-lines 5 --min-tokens 70 --report "build//phpcpd.xml"

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/parallel-lint" --exclude vendor -e php "src"

[Edge\QA\Task\ParallelExec] "/Users/shengyou/.composer/vendor/edgedesign/phpqa/../../../../vendor/bin/phpcs" -p --standard=PSR2 --ignore=*/vendor/* "src" --extensions=php --report-checkstyle="build//checkstyle.xml" exited with code 2

Time 0.186s

[Edge\QA\Task\ParallelExec] Exit code 2 Time 0.186s

[phpqa]

Tool	Allowed Errors	Errors count	Is OK	Report
phpmetrics			✓	build//phpmetrics/index.html
phploc			✓	build//phploc.html
phpcs		1	✓	build//phpcs.html
phpmd		0	✓	build//phpmd.html
pdepend			✓	build//pdepend.html
phpcpd		0	✓	build//phpcpd.html
parallel-lint		0	✓	build//parallel-lint.html
phpqa		1	✓	build//phpqa.html

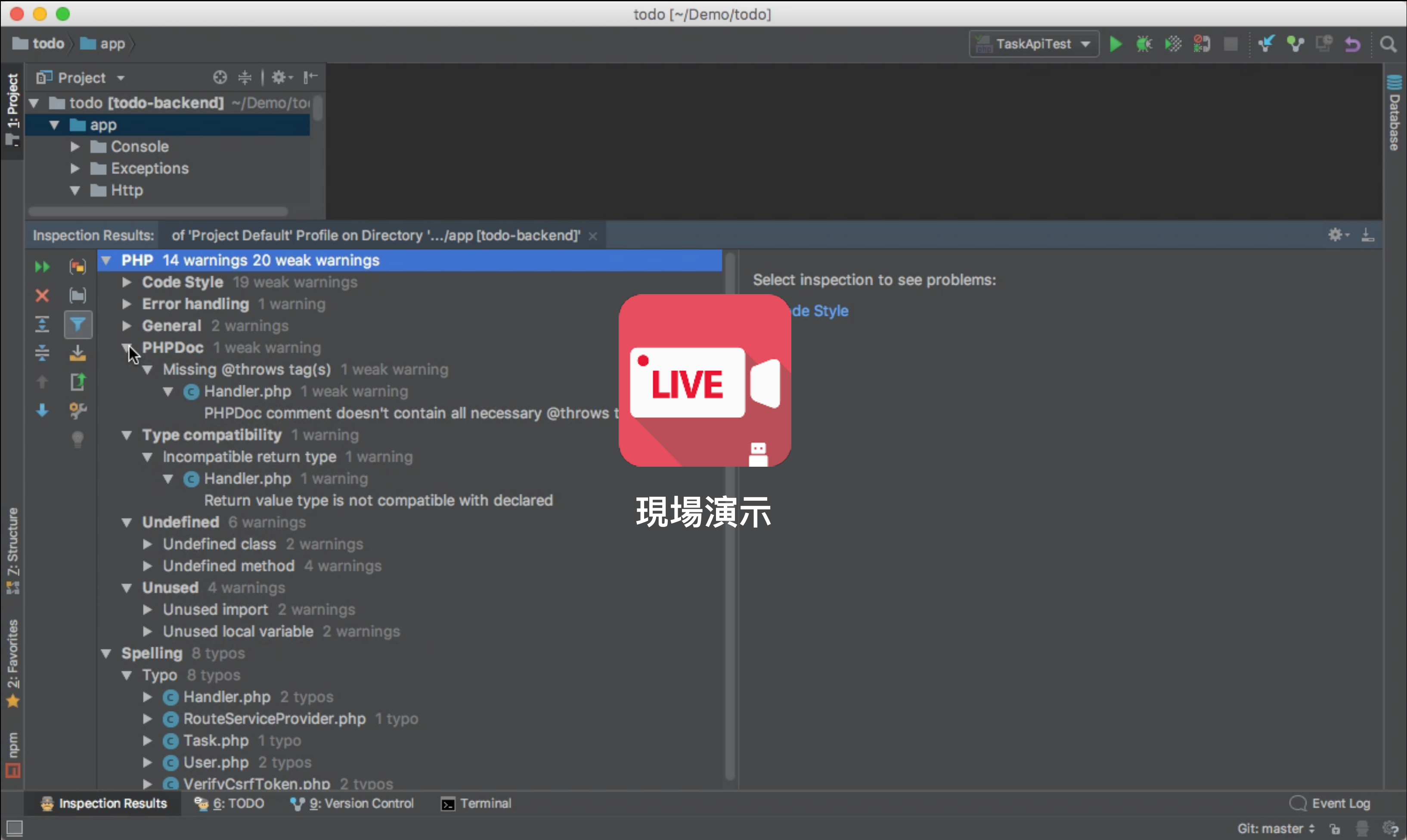
[phpqa] No failed tools

~/Demo/app master

現場演示

Code Inspection

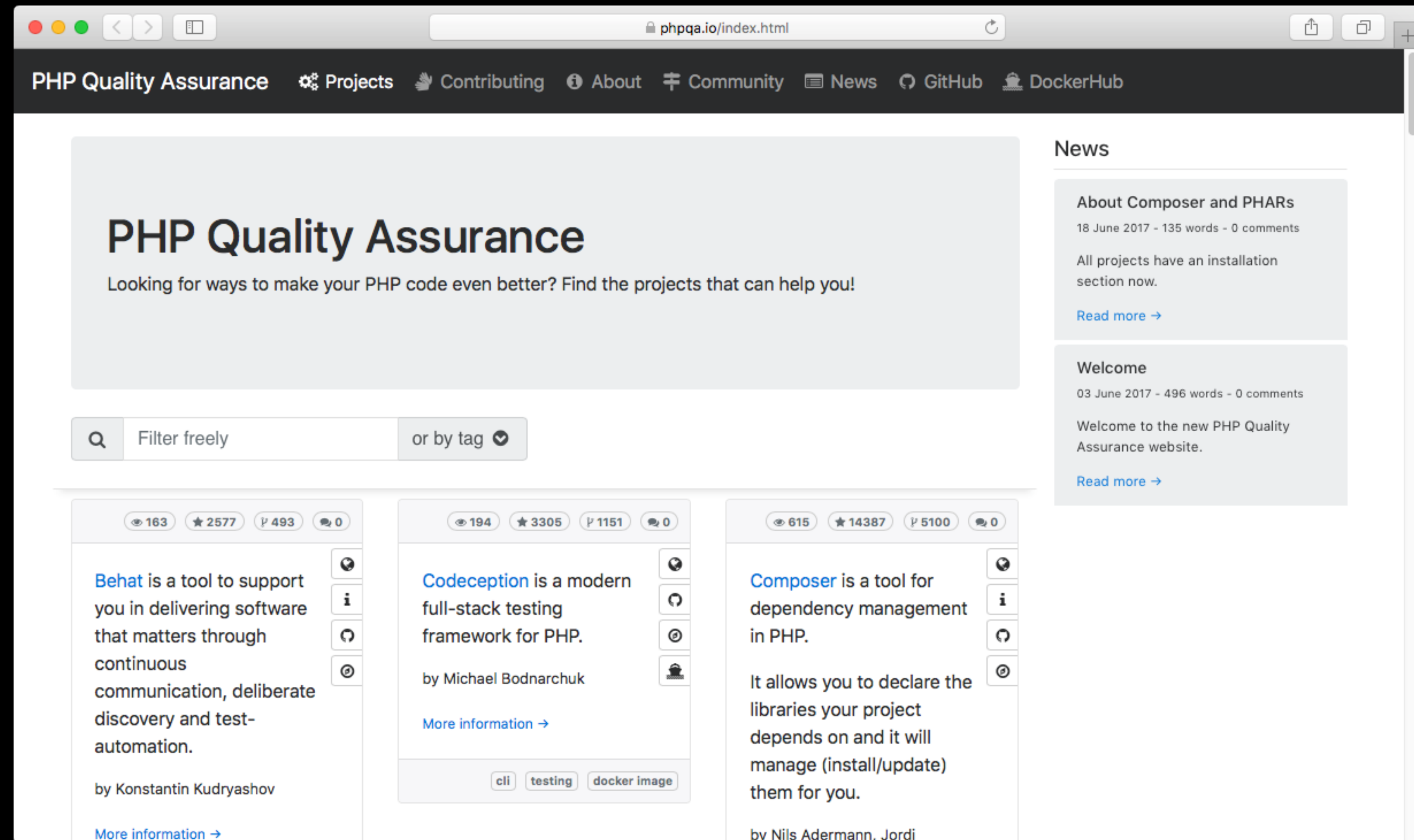
透過 Code Inspection 可以檢查出語法有潛在問題之處。



PHP Quality Assurance

這個網站專門列出各種跟 PHP Quality Assurance 有關的工具資料，可不定期來找找有無適合的工具。

<https://phpqa.io/>



Awesome PHP

在這個清單裡，不僅有與 Code Quality、Static Analysis 有關的相關函式庫，也有眾多與 PHP 開發相關的工具資料。

<https://github.com/ziadoz/awesome-php>

GitHub, Inc. github.com/ziadoz/awesome-php

Features Business Explore Marketplace Pricing This repository Search Sign in or Sign up

ziadoz / awesome-php Watch 1,727 Star 18,703 Fork 3,714

<> Code Issues 1 Pull requests 130 Projects 0 Insights

A curated list of amazingly awesome PHP libraries, resources and shiny things.

php php-framework php-library php-applications php-installation php7 awesome-lists awesome

1,313 commits 2 branches 0 releases 209 contributors WTFPL

Branch: master New pull request Find file Clone or download

MadalinTomescu and josegonzalez Update README.md (#904) Latest commit 87ea803 on Mar 24

.travis.yml	[ci] white list vagrantup.com, 3v4l.org, voicesoftheelephpant.com (#769)	a year ago
.varci.yml	Enable Var.CI	2 years ago
CODE-OF-CONDUCT.md	Add email address. Closes #623	2 years ago
COLLABORATING.md	Clarify in guidelines	2 years ago
CONTRIBUTING.md	Fix typo in CONTRIBUTING	5 months ago
LICENSE.md	Make license file markdown	4 years ago
README.md	Update README.md (#904)	2 months ago

README.md

測試開發成果

代碼真的如我所想的運作嗎？

- Unit Test
- Feature Test
- Browser Test
- API spec and testing

Unit Test

—

我所寫的 Class 裡的各 method 都有正確的行為嗎？

```
# 實際的 Class
class Linker
{
    public function autolink($text)
    {
        // ...
        return ...
    }
}
```

Unit Test

—

我所寫的 Class 裡的各 method 都有正確的行為嗎？

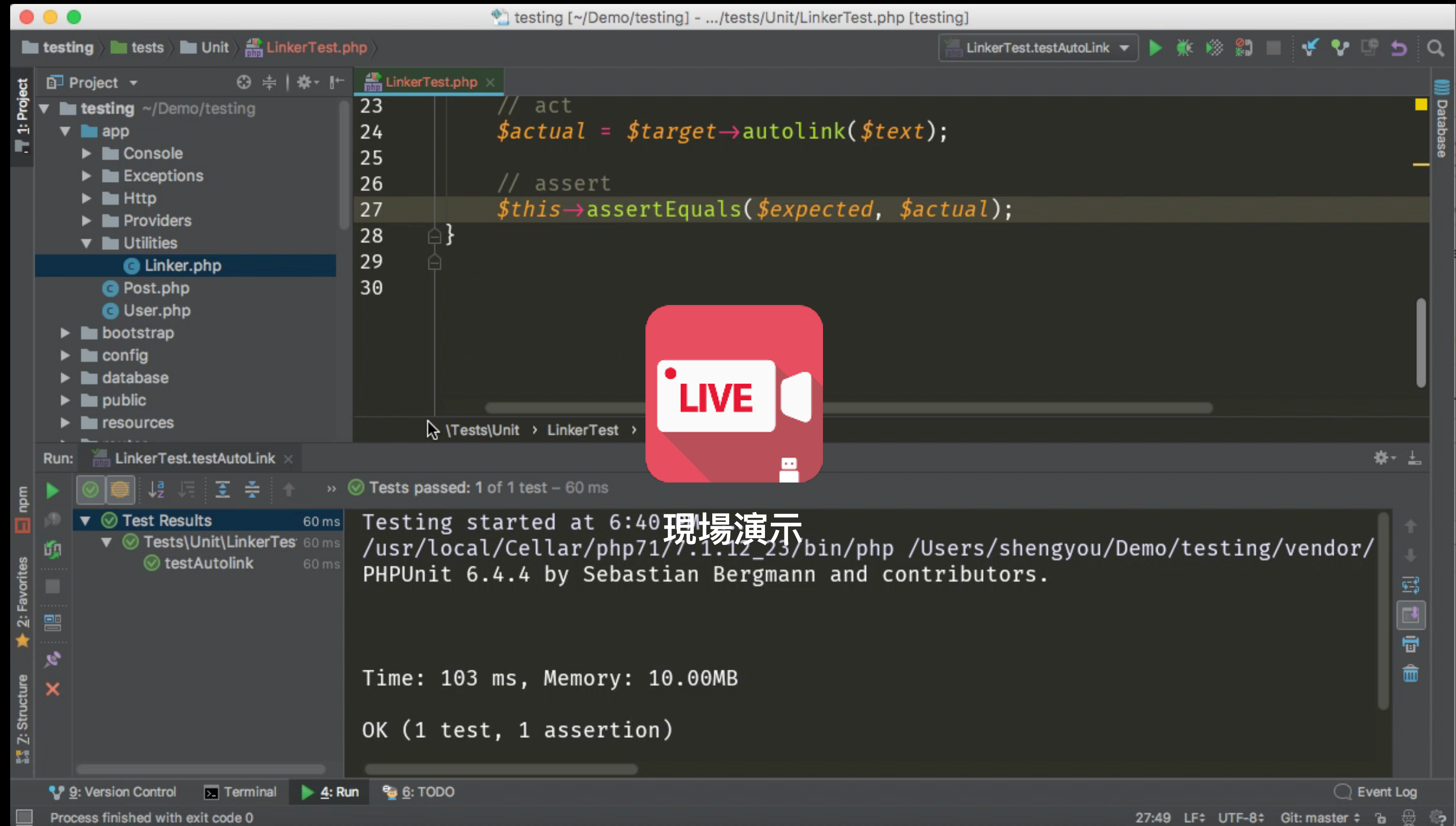
```
# 測試的 Class
class LinkerTest extends TestCase
{
    public function testAutolink()
    {
        // arrange
        $text = '...';
        $target = new Linker();
        $expected = '...';

        // act
        $actual = $target->autolink($text);

        // assert
        $this->assertEquals($expected, $actual);
    }
}
```

執行 Unit Test

從 Unit Test 通過與否來判定代碼運作是
否符合需求。



The screenshot displays an IDE interface with the following components:

- Project Explorer:** Shows a project structure with folders like 'app', 'Console', 'Exceptions', 'Http', 'Providers', 'Utilities', and files like 'Linker.php', 'Post.php', and 'User.php'.
- Code Editor:** Displays the content of 'LinkerTest.php' with line numbers 23 to 30. The code includes comments for 'act' and 'assert' phases, and a call to `$this->assertEquals($expected, $actual);`.
- Run Panel:** Shows the execution of `LinkerTest.testAutoLink`. The status bar indicates 'Tests passed: 1 of 1 test - 60 ms'.
- Test Results:** A tree view showing the test results for 'Tests\Unit\LinkerTest' and 'testAutolink', both marked as passed with a duration of 60 ms.
- Terminal:** Displays the output of the test run, including the start time (6:40), the PHP command used, the PHPUnit version (6.4.4), and the final status: 'Time: 103 ms, Memory: 10.00MB' and 'OK (1 test, 1 assertion)'.

A red 'LIVE' video icon is overlaid on the right side of the IDE window. The text '現場演示' (Live Demonstration) is written in Chinese characters over the terminal output.

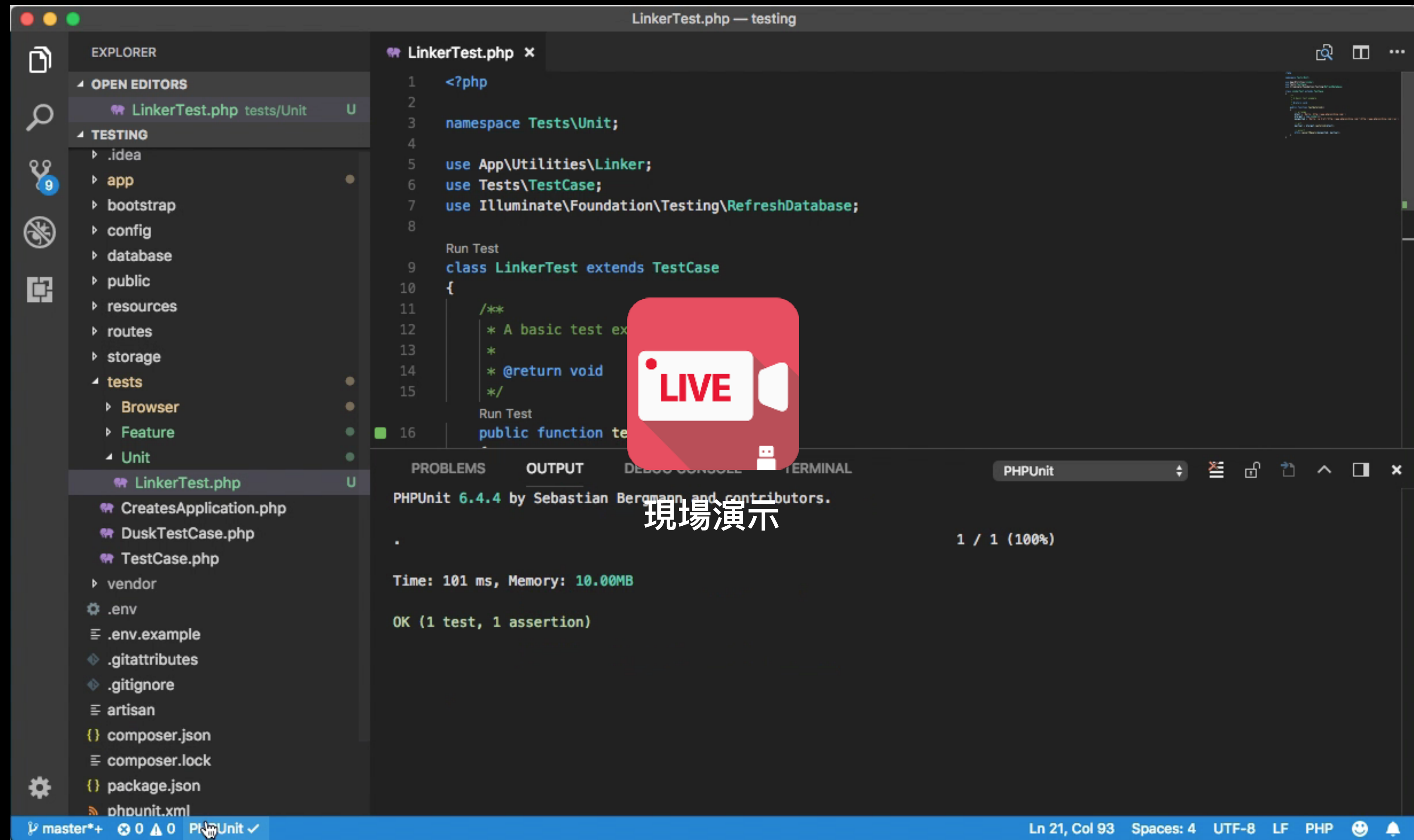
來自社群朋友的火力支援

Laravel 台灣社群朋友開發的 PHPUnit 外掛，讓執行測試的方式更簡潔快速又清楚易懂。

搜尋關鍵字：VSCode PHPUnit

作者關鍵字：Recca Tsai

<https://marketplace.visualstudio.com/items?itemName=recca0120.vscode-phpunit>



Feature Test

—

程式的運作邏輯如我想像的一樣執行嗎？

```
# 實際的 Class
class PostController extends Controller
{
    public function store(Request $request)
    {
        Post::create($request->all());

        return redirect()->route('...')
            ->with('message', 'Success');
    }
}
```

Feature Test

—

程式的運作邏輯如我想像的一樣執行嗎？

```
# 測試的 Class
class PostControllerTest extends TestCase
{
    use RefreshDatabase;
    use WithoutMiddleware;

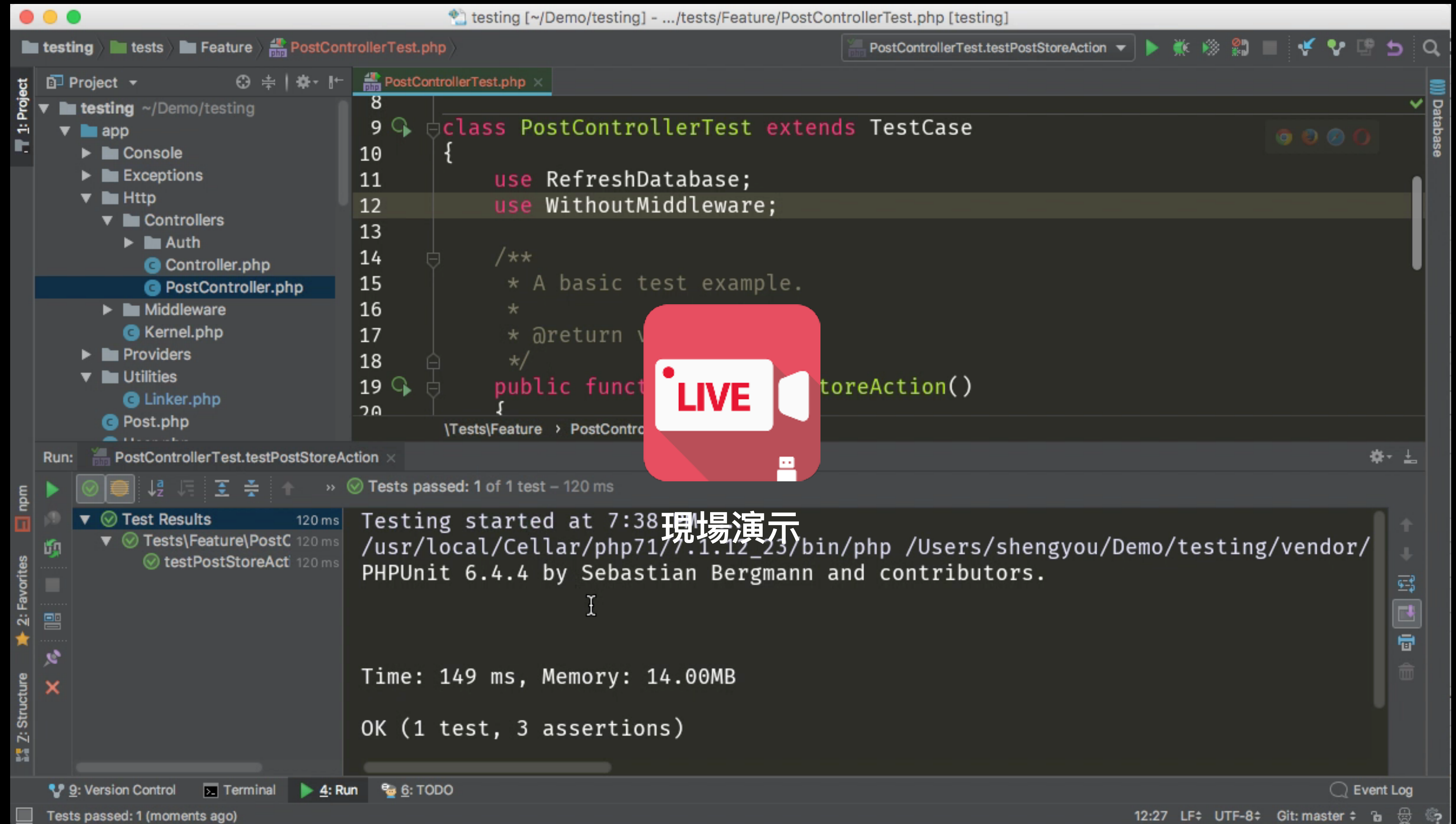
    public function testPostStoreAction()
    {
        $data = [
            'title' => '...',
            'content' => '...',
        ];

        $response = $this->post('...', $data);

        $response->assertRedirect('...');
        $this->assertDatabaseHas('...', $data);
    }
}
```

執行 Feature Test

從 Feature Test 通過與否來判定程式碼運作是否符合需求。



The screenshot displays an IDE window titled "testing [~/Demo/testing] - .../tests/Feature/PostControllerTest.php [testing]". The editor shows the following PHP code:

```
8
9 class PostControllerTest extends TestCase
10 {
11     use RefreshDatabase;
12     use WithoutMiddleware;
13
14     /**
15      * A basic test example.
16      *
17      * @return void
18      */
19     public function testPostStoreAction()
20     {
```

The interface includes a "Project" sidebar on the left showing the file structure, a "Run" toolbar at the top right, and a "Test Results" panel at the bottom. The "Test Results" panel shows a single test passing:

- Test Results: 120 ms
- Tests\Feature\PostC: 120 ms
- testPostStoreAct: 120 ms

The "Run" panel at the bottom displays the following output:

```
Testing started at 7:38
/usr/local/Cellar/php71/7.1.12_23/bin/php /Users/shengyou/Demo/testing/vendor/PHPUnit 6.4.4 by Sebastian Bergmann and contributors.

Time: 149 ms, Memory: 14.00MB

OK (1 test, 3 assertions)
```

A red "LIVE" badge is overlaid on the code editor, and the text "現場演示" (Live Demonstration) is written in the center of the image.

Browser Test

—

整合 headless 瀏覽器套件，讓測試行為
可以在瀏覽器裡重現。

```
# 安裝 dusk 套件
```

```
$ composer require --dev laravel/dusk
```

```
# 初始化
```

```
$ artisan dusk:install
```

```
# 建立 Browser Tests 檔案
```

```
$ artisan dusk:make {test}
```

Browser Test

—

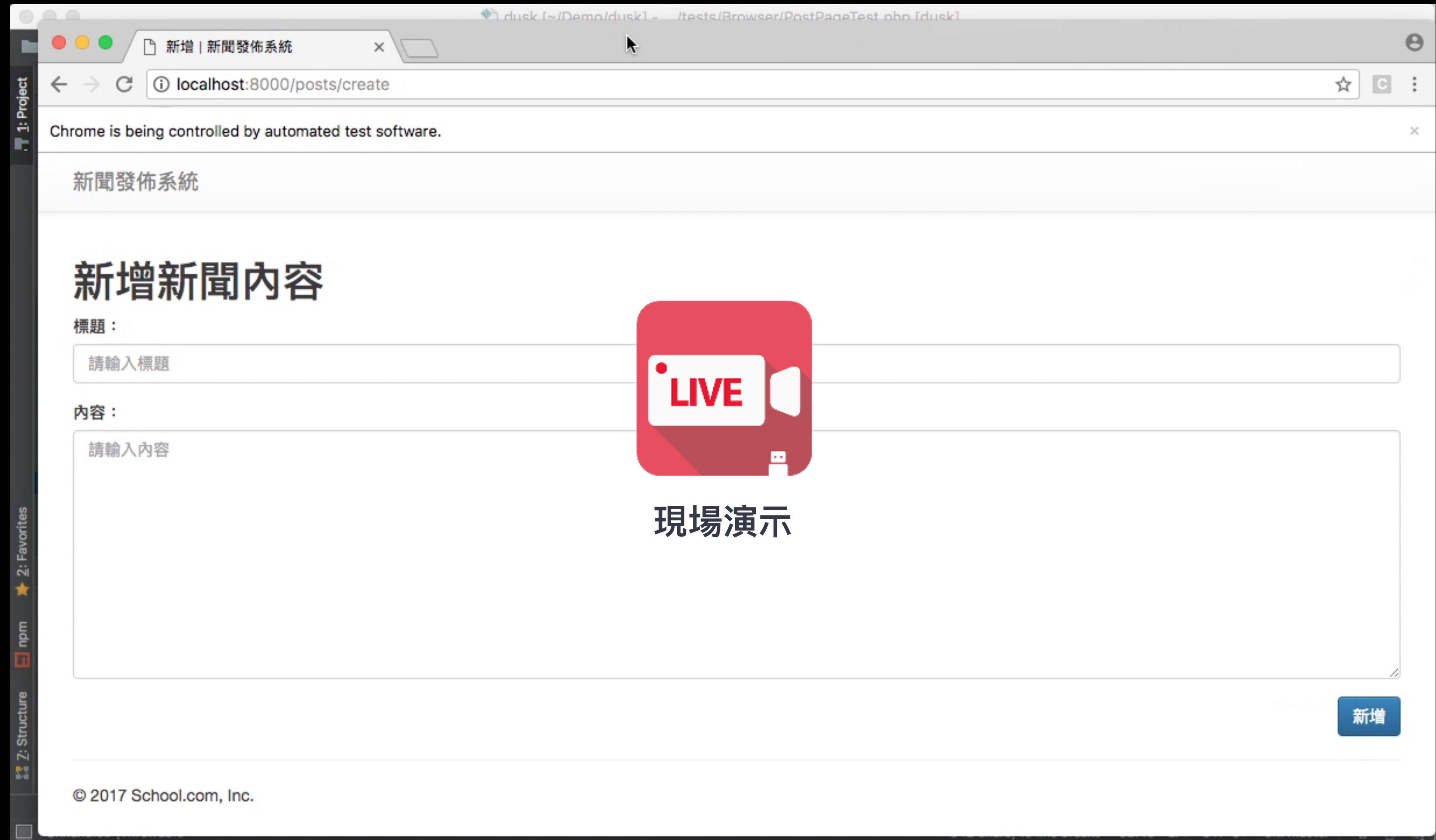
依照使用者實際的動作撰寫測試程式碼，
模擬使用者的操作行為。

模擬瀏覽動作

```
public function testSuccessSubmitForm()
{
    $this->browse(function (Browser $browser) {
        $browser->visit('...')
            ->assertSee('...')
            ->type('...', '...')
            ->pause(1000)
            ->press('...')
            ->assertPathIs('...')
            ->screenshot('...');
    });
}
```

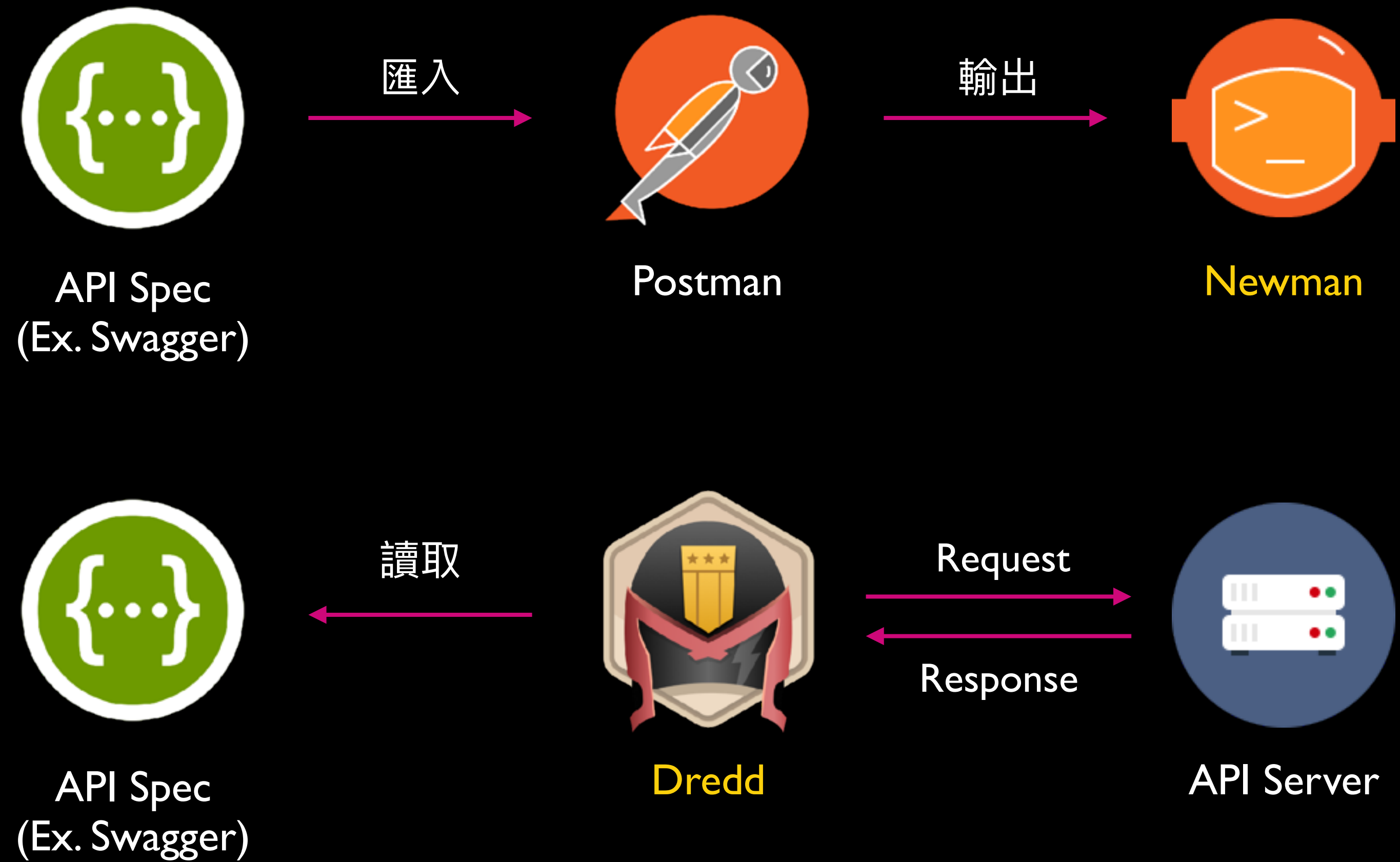
執行 Browser Test

使用者在瀏覽器裡看到的、體驗到的，
跟我預期的是一樣嗎？



API Spec and Testing

沒有畫面的 API，該怎麼定義 Spec 並測試它？

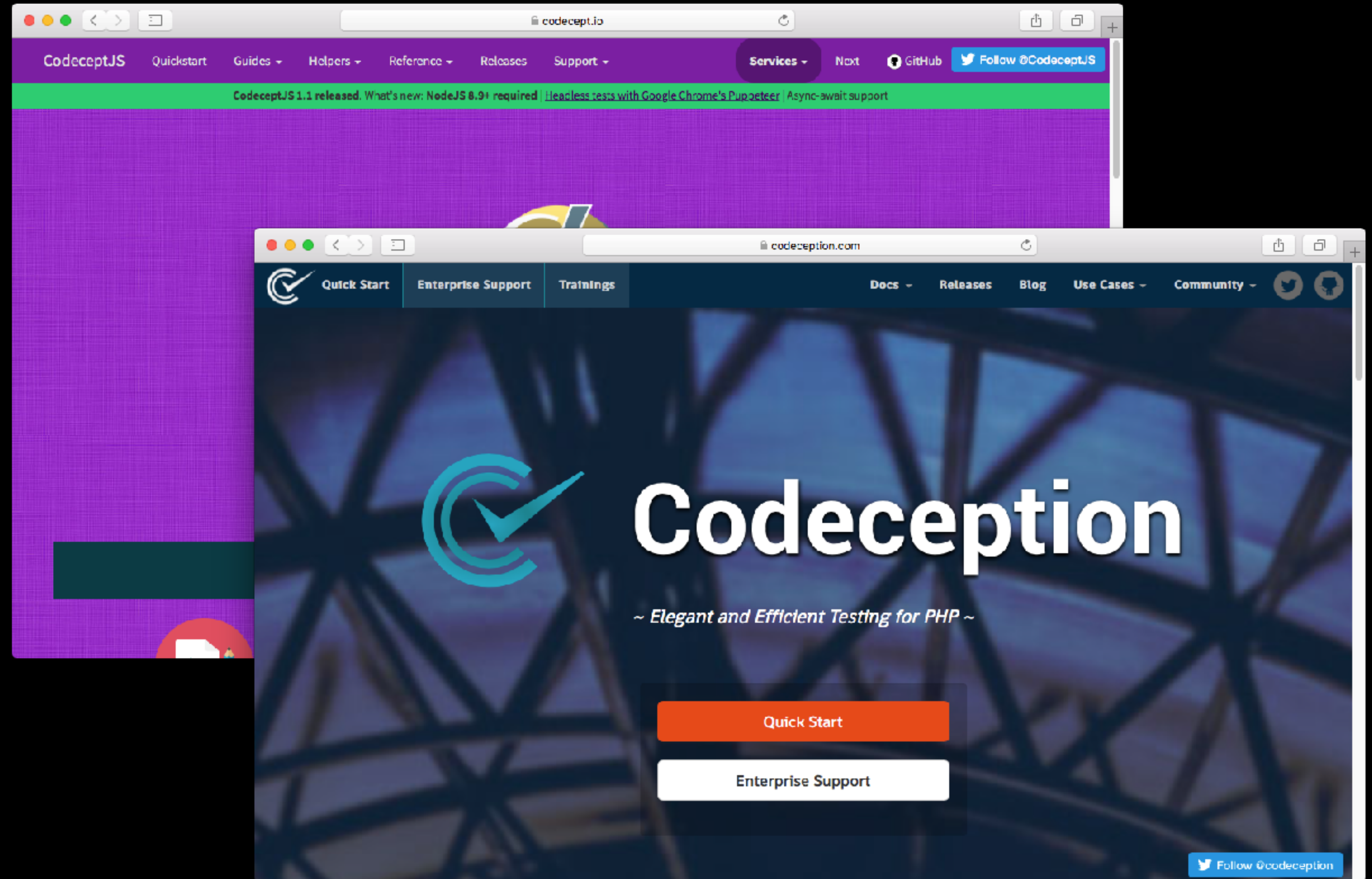


測試框架

測試框架百百種，挑一個適合、順手的
用：

<https://codeception.com>

<https://codecept.io>

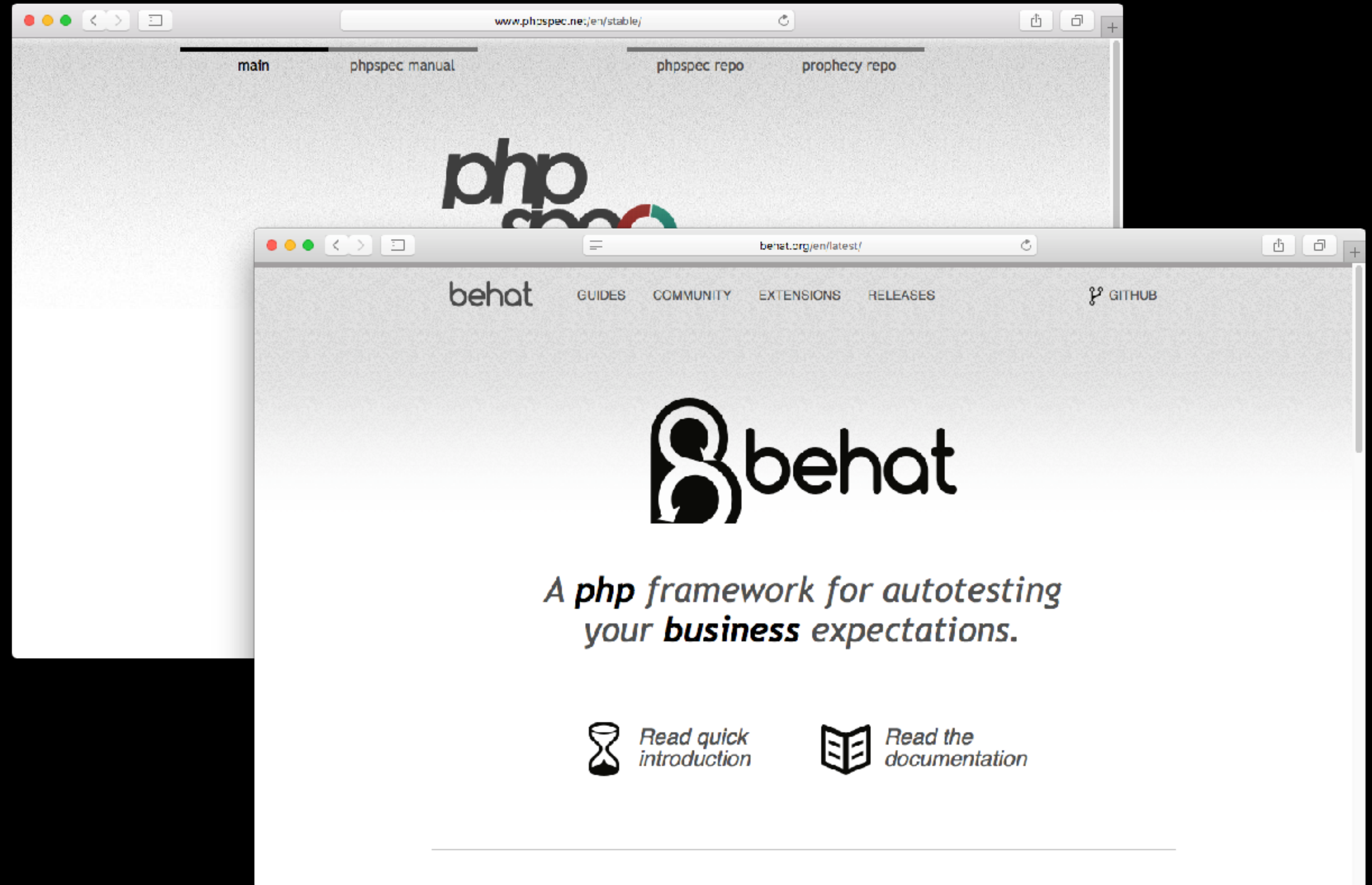


測試框架

測試框架百百種，挑一個適合、順手的
用：

<http://www.phpspec.net/en/stable/>

<http://behat.org/en/latest/>



步驟自動完成

該做的都做了，但忘了執行？試試自動化：

- Pre-commit hooks
- Continuous integration

Pre-commit hooks

攔截 VCS 提交事件，可以綁定工具來做自動化。

```
$ vim .git/hooks/{script}
```

<https://git-scm.com/docs/githooks>

<https://www.atlassian.com/git/tutorials/git-hooks>

```
1. shengyou@Taiji: ~/Demo/pre-commit (zsh)
~/Demo$ mkdir pre-commit
~/Demo$ cd pre-commit
~/Demo/pre-commit$ git init
Initialized empty Git repository in /Users/shengyou/Demo/pre-commit/.git/
~/Demo/pre-commit$ cd .git/hooks
~/Demo/pre-commit/.git/hooks$ ll
total 96
-rwxr-xr-x  1 shengyou  staff   478B May 14 01:29 applypatch-msg.sample
-rwxr-xr-x  1 shengyou  staff   896B May 14 01:29 commit-msg.sample
-rwxr-xr-x  1 shengyou  staff  3.2K May 14 01:29 fsmonitor-watchman.sample
-rwxr-xr-x  1 shengyou  staff   189B May 14 01:29 post-update.sample
-rwxr-xr-x  1 shengyou  staff   424B May 14 01:29 pre-applypatch.sample
-rwxr-xr-x  1 shengyou  staff   1.6K May 14 01:29 pre-commit.sample
-rwxr-xr-x  1 shengyou  staff   1.3K May 14 01:29 pre-push.sample
-rwxr-xr-x  1 shengyou  staff   4.8K May 14 01:29 pre-receive.sample
-rwxr-xr-x  1 shengyou  staff   544B May 14 01:29 pre-rotate.sample
-rwxr-xr-x  1 shengyou  staff   1.5K May 14 01:29 pre-sendemail.sample
-rwxr-xr-x  1 shengyou  staff   3.5K May 14 01:29 pre-tag.sample
~/Demo/pre-commit/.git/hooks$ cp pre-commit.sample pre-commit
~/Demo/pre-commit/.git/hooks$ vim pre-commit
~/Demo/pre-commit/.git/hooks$ ...
~/Demo/pre-commit$ vim test.txt
~/Demo/pre-commit$ git add .
~/Demo/pre-commit$ git commit -m "init commit"
Do something cool when pre commit
[master (root-commit) 2c6024e] init commit
1 file changed, 1 insertion(+)
create mode 100644 test.txt
~/Demo/pre-commit$
```



現場演示

grumphp

—

grumphp 套件支援多種 PHP 語法檢查工具，可在 pre-commit 時一併觸發。

<https://github.com/phpro/grumphp>



```
# 安裝 grumphp 套件
```

```
composer require --dev phpro/grumphp
```

```
# 設定 grumphp
```

```
# grumphp.yml
```

```
parameters:
```

```
    git_dir: .
```

```
    bin_dir: vendor/bin
```

```
tasks:
```

```
    phpcs: ~
```

```
    phpcsfixer2: ~
```

```
    phpcpd: ~
```

```
    phpmd: ~
```

```
    phan: ~
```

```
    phpunit: ~
```

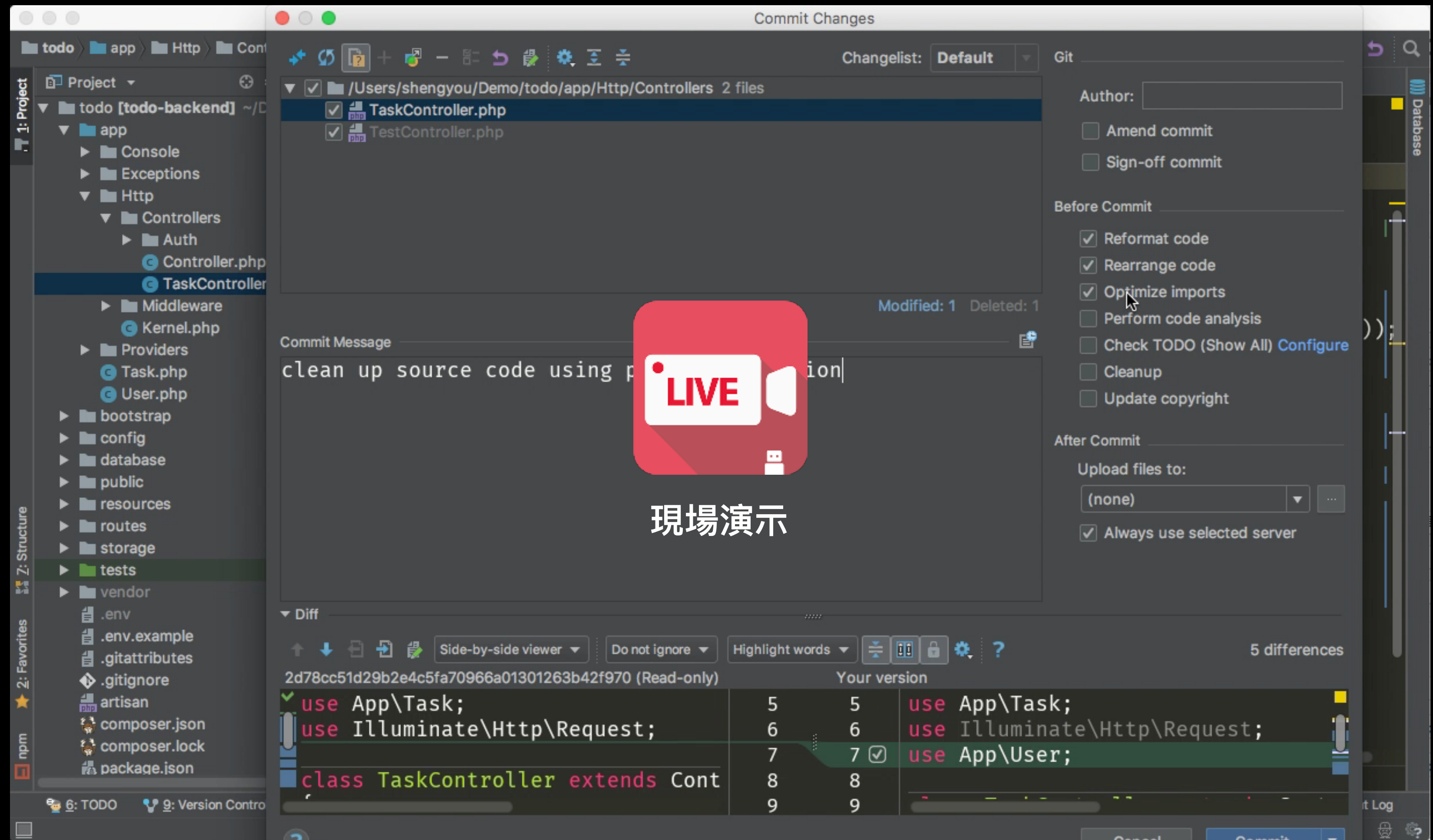
透過 grumphp 觸發

安裝 grumphp 套件後，每次於 git pre-commit 時一併觸發 phpcs、phpunit... 等工具。



透過 Editor/IDE 觸發

在自己順手的 Editor/IDE 裡觸發 VCS 的 pre-commit hooks，自動執行程式碼排版。



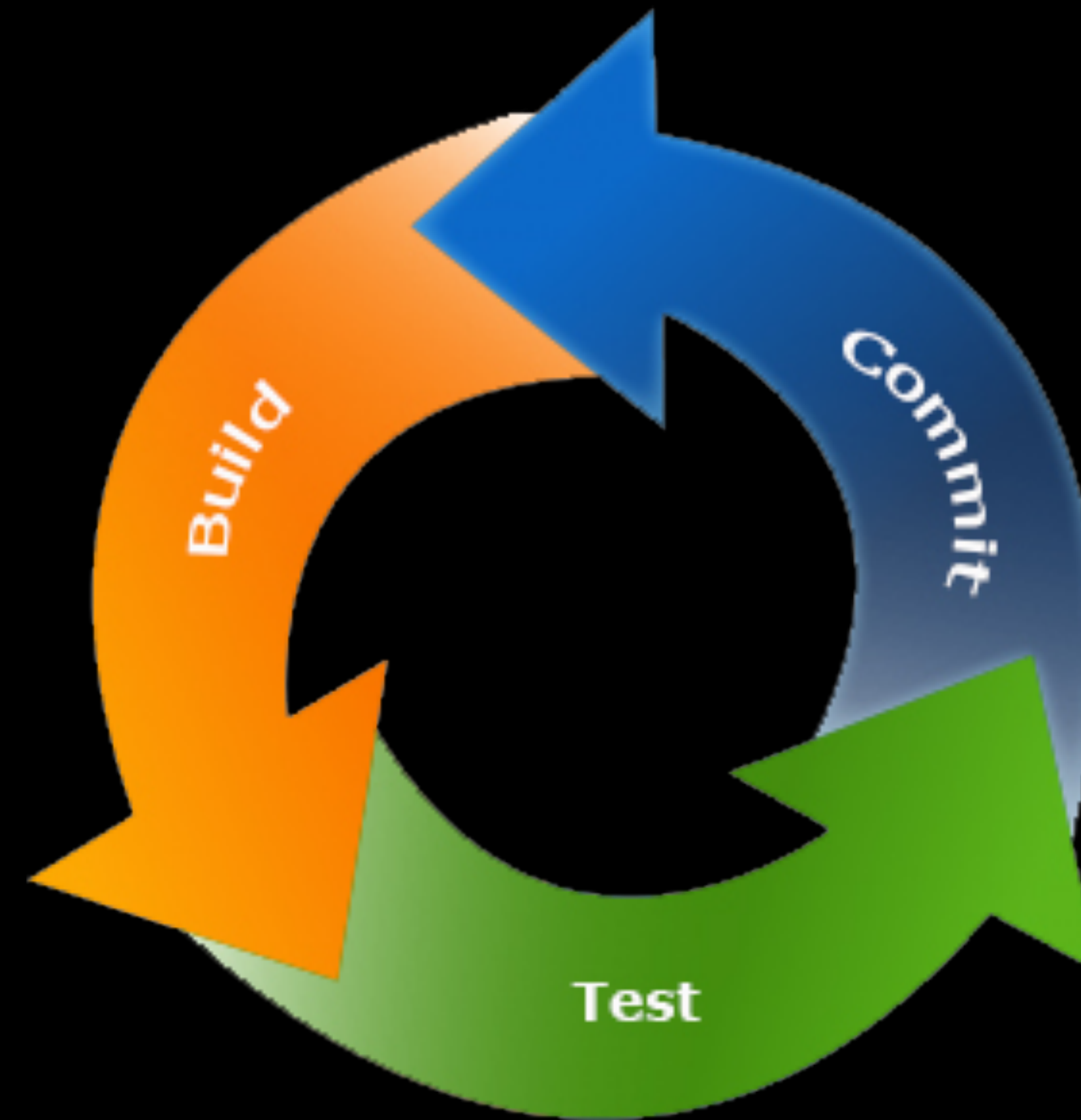
Continuous integration

可運用 CI 的情境：

- 不想/不需要在本機端跑的檢查/測試
- 本機端跑起來太花時間的檢查/測試
- 定期需要跑的檢查/測試

可以在 CI 上做的動作：

- 在 CI 上修正撰碼風格
- 在 CI 上檢查語法
- 在 CI 上產生報表

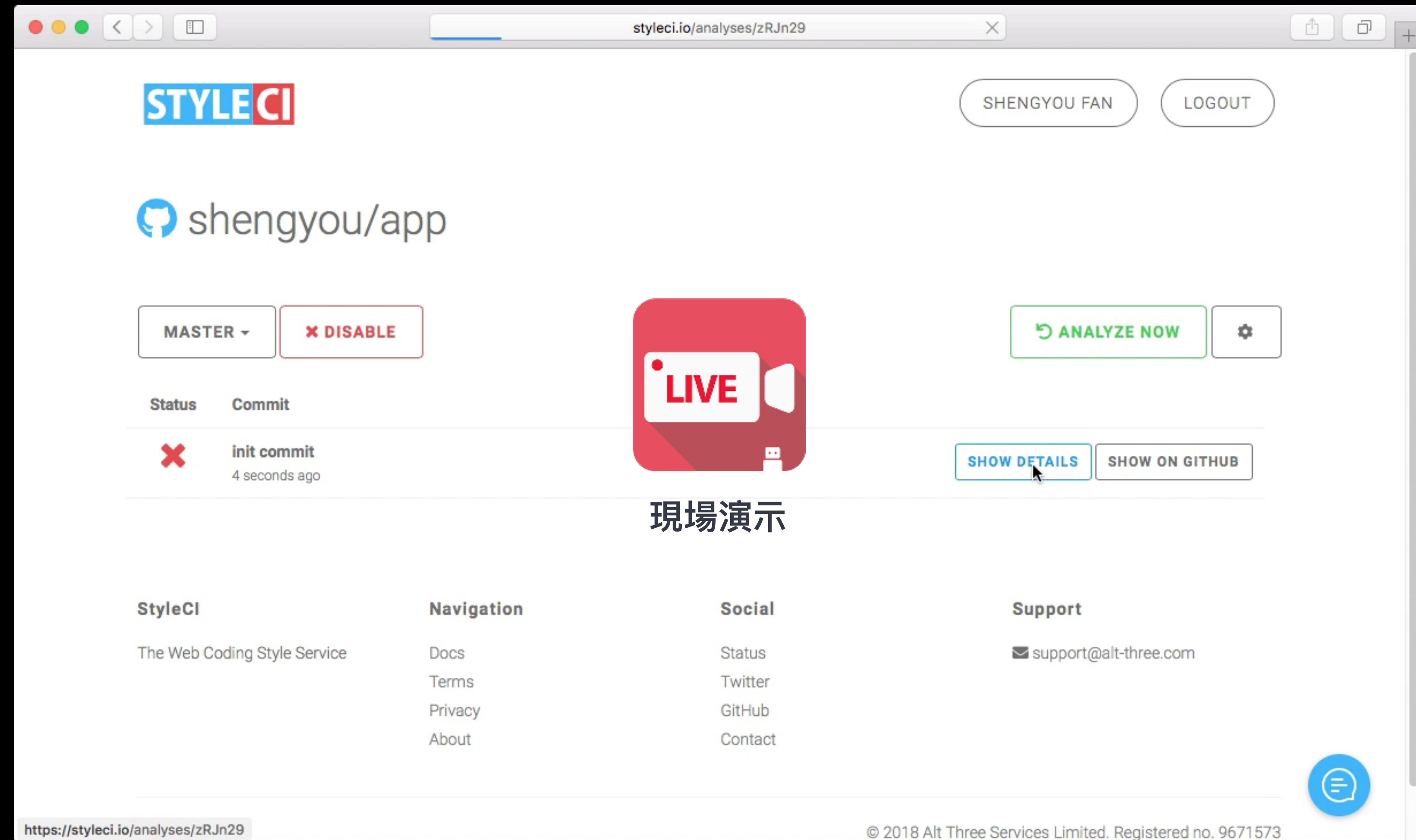


在 CI 上修正撰碼風格

使用 StyleCI 或 Nitpick CI 這一類的 CI 工具來修正撰碼風格。

當然，直接在 CI 上執行 phpdbf、php-cs-fixer 一類的工具也是可以的。

<https://styleci.io/>
<https://nitpick-ci.com/>

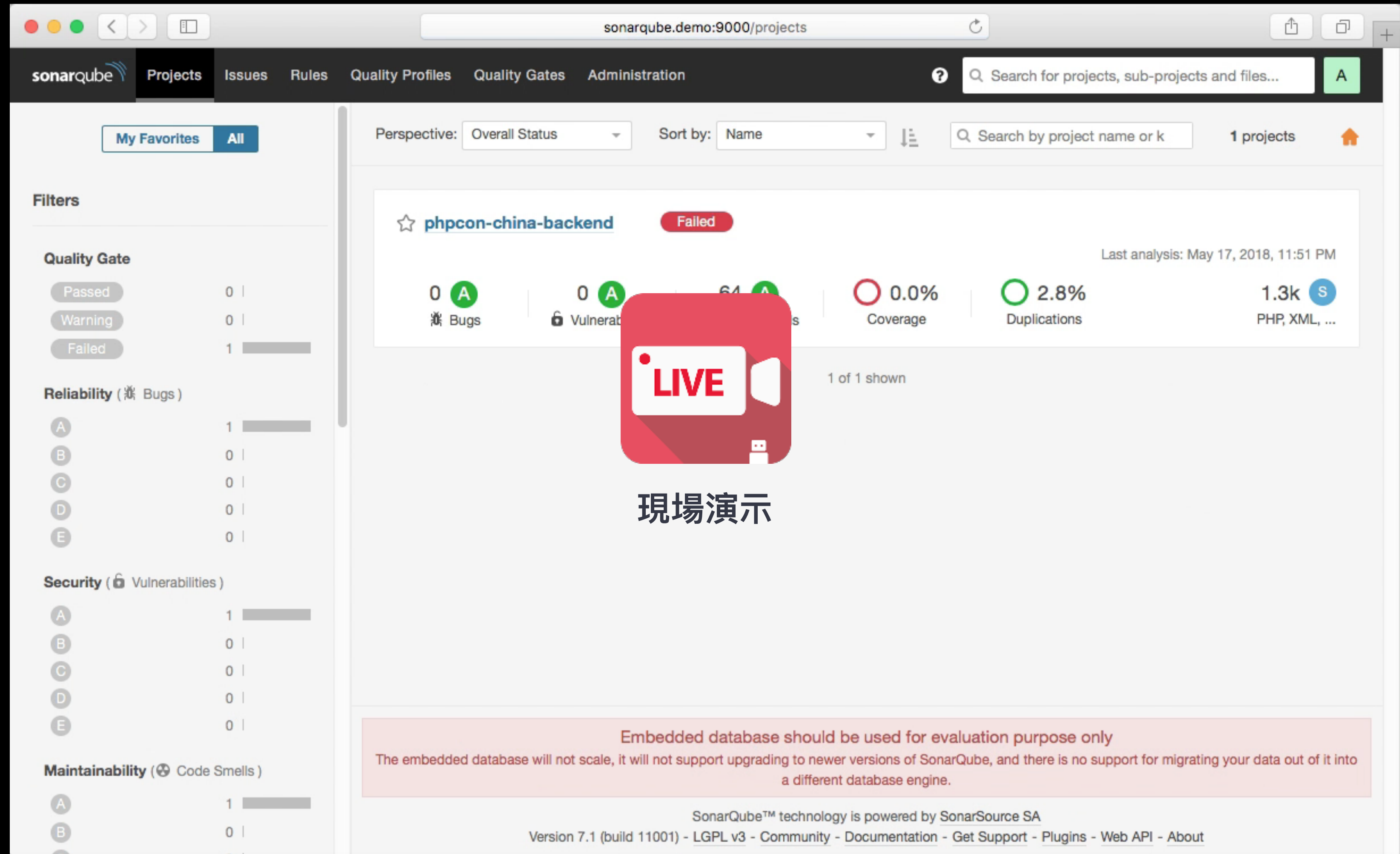


現場演示

在 CI 上檢查語法

使用 SonarQube 這一類的工具在 CI 主機上檢查語法。

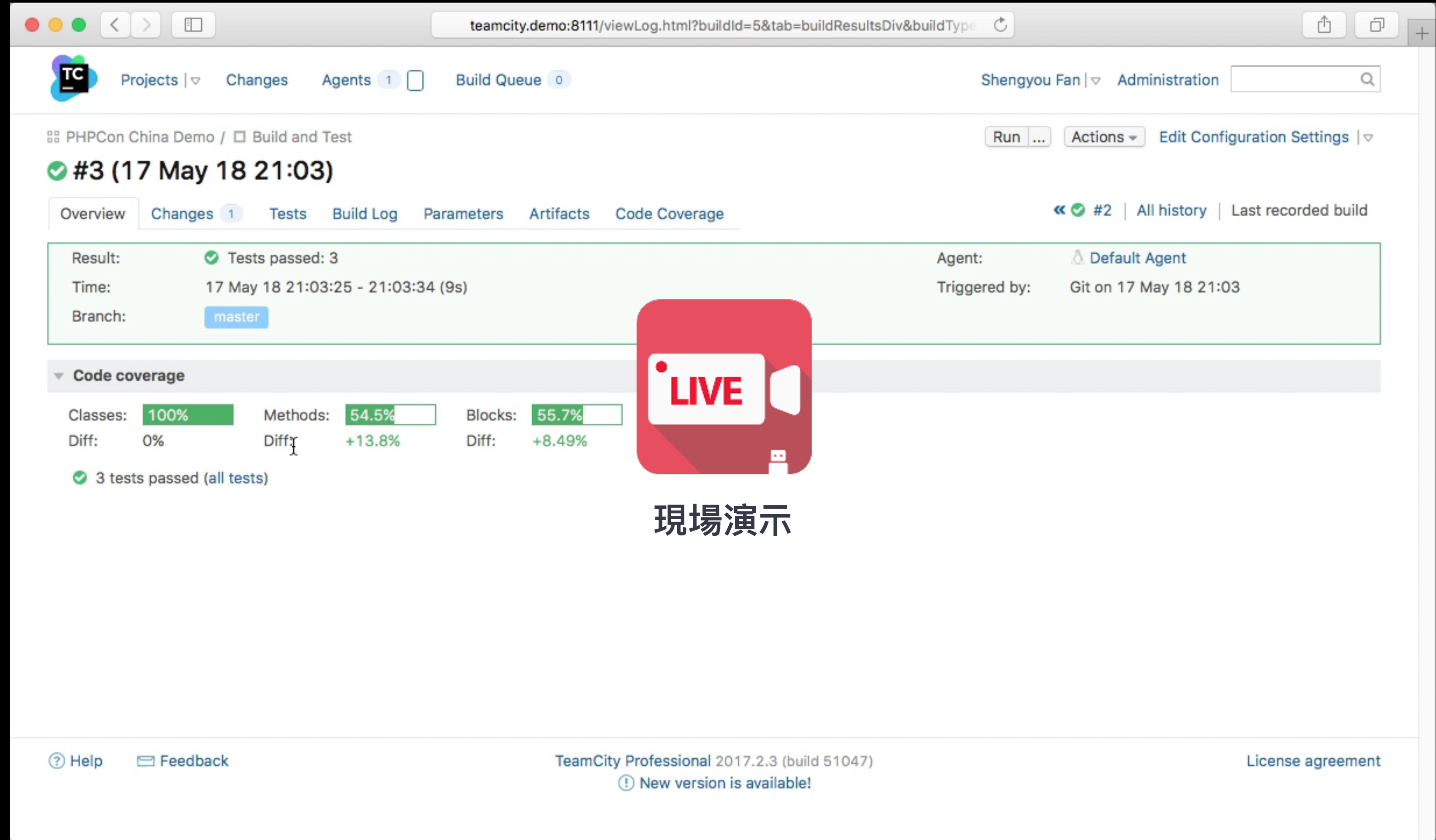
當然，直接在 CI 上執行 CodeSniffer、Copy/Paste Detector、Mess Detector 一類的工具也是可以的。



在 CI 上產生報表

一

在 CI 主機上產生代碼品質報表，供團隊評估程式碼品質趨勢。



The screenshot displays the TeamCity web interface for a build named "#3 (17 May 18 21:03)". The build status is "Success" (indicated by a green checkmark). The interface shows various tabs for build details, including Overview, Changes, Tests, Build Log, Parameters, Artifacts, and Code Coverage. The Code Coverage section is expanded, showing the following data:

Category	Value	Diff
Classes	100%	0%
Methods	54.5%	+13.8%
Blocks	55.7%	+8.49%

Below the code coverage data, it states "3 tests passed (all tests)". The interface also includes a "LIVE" video feed icon and a "現場演示" (Live Demo) label. The footer of the interface shows "TeamCity Professional 2017.2.3 (build 51047)" and a notification "New version is available!".

主題回顧

一



主題總結

—



沒有銀彈

沒有完美的工具、工具間各有特長

主題總結

—



最小施力點

依照團隊需要的，找出最適合的組合，專注在最有價值的事情。

主題總結

—



建立團隊共識

別忘了「人」才是團隊的主體，建立合作習慣與默契。

感謝聆聽
歡迎交流
—

關注作者：



<https://medium.com/@shengyou/>



<https://www.slideshare.net/shengyou/>



<https://www.youtube.com/user/shengyou/>