buzdykg 30 апреля 2015 в 14:53

Мини API на Lumen

Laravel, PHP

Из песочницы

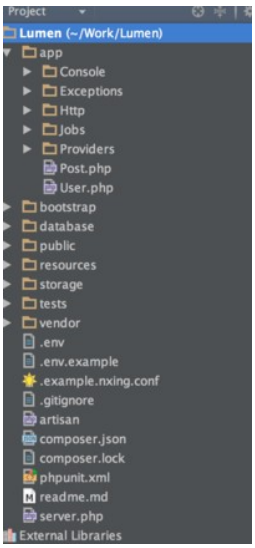
Цель этой публикации — создание простого API на Lumen и рассмотрение его отличий от старшего брата. Весь код доступен здесь.

Устанавливал я его на бокс homestead, т.ч. управился одной строчкой:

```
composer create-project laravel/lumen --prefer-dist Lumen
```

Подробнее о homestead.

Структура проекта похожа на Laravel:



Бросается в глаза отсутствие папки /config. Дело в том, что Lumen полностью полагается на содержимое .env файла. Пример содержимого:

```
DB_CONNECTION=mysql
DB_USERNAME=homestead
DB_PASSWORD=secret
DB_DATABASE=lumen
```

Все возможные настройки можно подглядеть в vendor/laravel/lumen-framework/config/.

ЧИТАЮТ СЕЙЧАС

Не купили DLC: функцию, которая спасла бы упавшие 737, «Боинг» продавал как опцию
90,6k 753

Компьютер Raspberry Pi встроили в клавиатуру для него
10,2k 40

Результаты Pwn2Own: Tesla Model 3 взломана, на ней поехал домой автор нового метода атаки
2,6k 0

Как я пишу конспекты по математике на LaTeX в Vim
32,1k 101

В личном чате Telegram можно удалять любые сообщения — даже чужие
17,2k 79

Игра для любителей и знатоков Linux
5,9k 4

Итак,

Структура БД

Допустим, нам нужна сущность пост, плюс 2 метода — регистрация и логин.
Для создания миграций воспользуемся командной утилитой artisan.

```
php artisan make:migration create_users_table
php artisan make:migration create_posts_table
```

Теперь в папке /database лежит 2 новых файла. В каждом по 2 метода, up и down — миграция и отмена миграции.
Так выглядит метод up у таблицы users:

```
//database/*create_users_table.php
Schema::create('users', function (Blueprint $table) {
    $table->increments('id');
    $table->string('email')->unique();
    $table->string('first_name');
    $table->string('last_name');
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
    $table->softDeletes();
});
```

Для постов:

```
//database/*create_posts_table.php
Schema::create('posts', function (Blueprint $table) {
    $table->increments('id');
    $table->string('user_id');
    $table->string('content');
    $table->timestamps();
    $table->softDeletes();
});
```

Миграции готовы, пытаемся их применить:

```
php artisan migrate
```

Но артисан почему-то не видит переменные из .env файла и жалуется на плохие параметры подключения.
Для того, чтобы экспортнуть переменные, нужно раскомментировать строчку в /bootstrap/app.php:

```
Dotenv::load(__DIR__.'/../');
```

Также по-умолчанию выключен ORM Eloquent и фасады. Штуки полезные, поэтому я их тоже включил.
Теперь всё должно получится:

```
vagrant@homestead:/www/Lumen$ art migrate
Migrated: 2015_04_26_210913_create_users_table
Migrated: 2015_04_26_210931_create_posts_table
```

(art — это alias для php artisan)

Также создадим Eloquent модели для этих таблиц. Например, модель поста:

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

class Post extends Model
{
    /**
     * @return \Illuminate\Database\Eloquent\Relations\BelongsTo
     */
    public function user()
```

```
    {
        return $this->belongsTo('Written\Models\User');
    }
}
```

Модели позволяют с меньшей болью отдавать данные с методов, т.к. они заботятся о взаимосвязи таблиц. Конечно, производительность «сырых» запросов к бд лучше, но скорость разработки при таком подходе будет неуклонно деградировать. Такие запросы уместны, на мой взгляд, только для статистических выборок.

Контроллеры

В Laravel 5 есть замечательный Trait, который позволяет сделать всю регистрацию в два щелчка пальцев. К сожалению в Lumen такого нет. Также сейчас принято не записывать все роуты в один файл, а пользоваться аннотациями, например:

```
/**
 * @Middleware("auth.token")
 * @Resource('post')
 */
class PostsController extends Controller {
    public function index() {}
    public function show($id) {}
    public function store() {}
    public function update() {}
    public function destroy() {}
}
```

Данная аннотация говорит, что контроллер RESTful. Т.о. имея перед глазами 1 открытый файл уже есть понимание, как обращаться к методам, и что за фильтры они имеют. Делается это с помощью либы laravelcollective/annotations. Но с Lumen она несовместима, поэтому все роуты придется пихать в /app/http/routes.php:

```
$app->get('/', function() use ($app) {
    return $app->welcome();
});
$app->post('/register', ['uses' => 'App\Http\Controllers\AuthController@postRegister']);
$app->post('/login', ['uses'    => 'App\Http\Controllers\AuthController@postLogin']);

$app->get('/post/{id}', ['uses' => 'App\Http\Controllers\PostsController@show']);
$app->get('/post',      ['uses' => 'App\Http\Controllers\PostsController@index']);
$app->group(['middleware' => 'logged.in'], function($app) {
    $app->post('/post',     ['uses' => 'App\Http\Controllers\PostsController@store']);
    /** & another protected routes */
});
```

В нормальном приложении этот файл становится монструозным быстро.

У Lumen, как и у Laravel, есть Middleware, которые могут либо фильтровать определенные запросы, либо делать всякие полезности для каждого запроса. Все такие фильтры лежат в /app/Http/Middleware/. Для того, чтобы Lumen знал об их существовании, нужно добавить соответствующие классы в /bootstrap.app.php.

Пример Middleware:

```
//app/Http/Middlewared/LoggedInMiddleware.php
class LoggedInMiddleware {
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  \Closure  $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if(!Auth::check()) {
            return new Response('', 401);
```

```
        }

        return $next($request);
    }
}
```

Описанный фильтр выдаёт http код 401, если запрос исходит от неавторизованного пользователя.

Пример контроллера:

```
//app/Http/Controller/PostsController.php
/**
 * Class PostsController
 * @package App\Http\Controllers
 */
class PostsController extends Controller
{
    public function index()
    {
        return $this->respondWithData(Post::with('user')->all()->toArray());
    }

    public function show($id)
    {
        return $this->respondWithData(Post::find($id)->with('user')->get()->toArray());
    }

    public function store()
    {
        $rules = [
            'text' => 'required',
        ];

        $input = $_POST;

        $validator = Validator::make($input, $rules);
        if ($validator->fails()) {
            return $this->respondWithFailedValidation($validator);
        }

        $post = new Post;
        $post->content = $input['content'];
        $post->user()->associate(Auth::user());
        $post->save();

        return $this->show($post->id);
    }

//    public function delete() {}
}
```

Пример профита от использования Eloquent можно увидеть в методе show(). Клиенту отдаётся не только информация о посте, но также об ассоциированном пользователе.

respondWith* методы — вспомогательные, для придания коду какой-то организованности. В целом метод может возвращать даже обычную строку.

Заключение

Не зря заявлено, что Lumen полностью совместим с Laravel, т.к. после всего написанного я не чувствую, что написал что-то про Lumen.

Но всё же при разработке даже вышеописанного функционала остался осадочек:
— несовместим с библиотеками, написанными для Laravel. Те же аннотации де-факто стандарт;
— для вхождения нужно знать Laravel, т.к. многое, описанное в доках Laravel не работает, а в доках Lumen написано мало. Приходится смотреть исходники. Например фасады — доступно далеко не все. Недостающие нужно регистрировать самому, ручками;
— завести тесты у меня не удалось, т.к. почему-то в метод не прилетает $_POST.

У меня только ~~один~~два вопроса — зачем нужен Lumen, когда есть Laravel? Разве есть люди, которые хотят мегапроизводительности и при этом не пишут своё решение?

Теги: php laravel lumen

+4 66 24k 19

7,0 Карма 0,0 Рейтинг 1 Подписчики

@buzdykg
Пользователь

Поделиться публикацией

ПОХОЖИЕ ПУБЛИКАЦИИ

27 мая 2016 в 16:02
Настройка Laravel relationships — подсчет комментариев (вольный перевод)
+9 8,3k 69 16

13 марта 2016 в 23:32
PHP-Дайджест № 81 – интересные новости, материалы и инструменты (1 – 13 марта 2016)
+33 19,4k 103 11

9 июня 2015 в 21:33
Laravel 5.1
+16 23,1k 61 2

ВАКАНСИИ **Мой круг**

PHP Laravel разработчик — от 800 до 1500 $
Оки-Токи: Колл-центр в облаках • Возможна удаленная работа

Веб-разработчик Laravel — от 80000 ₽
INVITE Transport Software • Тольятти • Возможна удаленная работа

PHP-программист в Зеленограде (Yii2, Laravel) — от 70000 до 110000 ₽
Студия Валерия Комягина • Москва

PHP-программист — от 50000 ₽
CENTRA • Новокузнецк

Laravel/Yii2 разработчик на удаленку — от 60000 до 150000 ₽
Риверстарт • Нижний Новгород • Возможна удаленная работа

Все вакансии

Комментарии 19

Fesor 30 апреля 2015 в 14:57 +1

> Разве есть люди, которые хотят мегапроизводительности и при этом не пишут своё решение?

ЧТО ОБСУЖДАЮТ

Сейчас Вчера Неделя

Upwork регистрируется в РФ
4,2k 14

Да, есть. Если можно что-то не писать — лучше не писать. По сути Lumen предоставляет вам самый базовый костяк для того что бы делать что-то свое.

p.s. Осталось только сделать микро-фреймворк на базе Symfony/HttpKernel + HttpFoundation (или на PSR-7), FastRoute, Doctrine и PHP-DI.

crmMaster 30 апреля 2015 в 16:18 **–2**

Или просто переходить на Ruby. Там есть темы и синатра.

 **andrewnester** 30 апреля 2015 в 16:18 **0**

Лично мое мнение — микрофреймворк и доктрина это как муха и слон

И второе — HttpKernel + HttpFoundation это ли не Silex?

 **Fesor** 30 апреля 2015 в 16:58 **0**

Ну ок, Silex — Symfony/routing — Pimple + php-di + FastRoute. Согласитесь — это уже не Silex.

> Лично мое мнение — микрофреймворк и доктрина это как муха и слон

смотря зачем вам микрофреймворк. Меня вот концепция микрофреймворков устраивает тем что можно быстро перелопатить и собрать на базе оного свой фреймворк под задачу. Ну а для кого-то это хорошая штука для быстрого прототипирования.

 **andrewnester** 30 апреля 2015 в 17:21 **0**

фреймворки, в том числе и микро, очень спорная тема — вот например зачем брать микрофреймворк и перелапачивать его под свои нужды, а не просто взять нужные библиотеки и использовать их?

> Ну ок, Silex — Symfony/routing — Pimple + php-di + FastRoute. Согласитесь — это уже не Silex.

зачем одновременно Pimple + php-di?
в silex к слову Pimple используется, другое дело если вы его на php-di поменять хотите

 **Fesor** 30 апреля 2015 в 18:53 **0**

> зачем одновременно Pimple + php-di?

так я же написал — минус pimple. Хабрапарсер и все такое.

> а не просто взять нужные библиотеки и использовать их?

если есть микрофреймворк который использует все что тебе надо и тебе надо только заменить один два компонента — почему бы и нет.

 **HunterNNm** 30 апреля 2015 в 15:45 **–1**

Не хочу разводить холивары… Но не проще ли API делать на Phalcon? Скорость работы у них на высоте, всё нужное под рукой. При необходимости к API можно прикрутить и весь остальной функционал. Количество кода не сильно отличается. Да и чтобы использовать Lumen нужно сначала изучить Laravel — а порог вхождения там не сильно низкий.

 **buzdykg** 30 апреля 2015 в 16:04 **0**

С phalcon не работал, но просмотрев это сравнение, полагаю они достойные соперники.

Замечу, что с денежной точки зрения Laravel выигрывает полностью. Работы на нем становится всё больше. Пруф — поиск работы по ключевым словам «Phalcon»/«Laravel» на oDesk дал 7 против 234.

Так что если выбор стоит какой фреймворк изучать, то я бы советовал Laravel. А потом и Lumen, если «припрёт».

 **HunterNNm** 30 апреля 2015 в 16:18 **0**

Про денежность не спорю, но я больше за скорость работы. С интересом наблюдаю за языком Zephir — еще можно будет под Phalcon писать модули, что тоже дает ему +. Поднимали 2 аналогичных проекта: 1 на Laravel, второй на Phalcon. Потом 1-й переписывали на Phalcon из-за медленной работы. Теперь держит 350к человек в сутки с минимальными задержками, в то время как Laravel ложил довольно неслабый сервер.

 **andrewnester** 30 апреля 2015 в 16:36 **+1**

Не купили DLC: функцию, которая спасла бы упавшие 737, «Боинг» продавал как опцию
90,6k 753

Как я пишу конспекты по математике на LaTeX в Vim
32,1k 101

Польские учёные напечатали первую в мире бионическую поджелудочную железу с сосудами
7,9k 27

В личном чате Telegram можно удалять любые сообщения — даже чужие
17,2k 79

для laravel использовался opcache? пробовали nginx+fpm?
просто что-то мне подсказывает, что вовсе не в laravel может быть дело, но я могу ошибаться — не видел проектов

HunterNNm 30 апреля 2015 в 19:26 0

Сервер работает без апача, php-fpm + nginx, opcache включен и использовался, всё на SSD… БД — MySQL, структура и запросы после перехода не менялись. Но вот скорость работы заметили существенную.

Fesor 30 апреля 2015 в 19:28 +1

Ну… вы ж не просто бросились сломя голову переписывать, это как-то не логично. Вы прогоняли через XProf/Blackfire?

Fesor 30 апреля 2015 в 17:01 0

> С интересом наблюдаю за языком Zephir

С интересом наблюдаю за языком Go, где все быстрее чем с зефиркой и без извращений. Вы же понимаете что это совсем другой стэк технологий? А еще есть Hack.

> Поднимали 2 аналогичных проекта: 1 на Laravel, второй на Phalcon.

Хм… то есть для вашего приложения дешевле было переписать приложение чем купить еще один-два сервера?

Вот честно, было бы круто иметь какое-то базовое приложение в духе блога, что бы не совсем уж hello world, с нагрузочными тестами, и реализовать его на разных фреймворках. Но это куча работы…

Я прекрасно понимаю профит который дает фалькон в плане производительности, но как по мне риски связанные с его использованию в 90% случаев не покрывают профит.

HunterNNm 30 апреля 2015 в 19:29 0

В штате есть 2 программиста, которые всё за месяц переписали. Покупка серверов стоила бы дороже. Да и результат получился очень даже хороший. А вот идея про тестирование фреймворков — это интересно. Если в ближайшее время не будет никаких срочных заданий попробуем написать по такому приложению на Laravel/Symfony/Yii/Phalcon. Для себя же полезно

Fesor 30 апреля 2015 в 20:37 0

> В штате есть 2 программиста, которые всё за месяц переписали.

Еще более интересно посмотреть что являлось узким местом приложения на самом деле. Судя по всему приложение не такое большое (либо все хорошо покрыто тестами и отделено от фреймворка).

Что до приложения, если займетесь поделитесь ссылочкой на репозиторий. Могу помочь с Symfony2/Silex. Еще было бы круто все это в Docker контейнерах…

Corpsee 30 апреля 2015 в 19:43 0

Есть вот такой репозиторий: github.com/Grafikart/BlogMVC (Phalcon-а только нет, к сожалению). Осталось добавить нагрузочных тестов.

Fesor 30 апреля 2015 в 16:14 +1

> Но не проще ли API делать на Phalcon

Нет, не проще. Целевая аудитория фреймворков разная, хотя у Phalcon тоже есть неплохой микрофреймворк.

andrewnester 30 апреля 2015 в 16:19 0

А если баг в самом Phalcon найдете да еще и в пятницу вечером и на продакшне, что делать будете?

HunterNNm 30 апреля 2015 в 19:30 0

Ну до продакшена всё обкатывается несколько дней + тестирование. Еще ни одного случая не было. тьфу-тьфу-тьфу

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

САМОЕ ЧИТАЕМОЕ

Сутки | Неделя | Месяц

Как я пишу конспекты по математике на LaTeX в Vim

+165 32,1k 410 101

Каким был первый айфон?

+43 33,1k 38 9

Не купили DLC: функцию, которая спасла бы упавшие 737, «Боинг» продавал как опцию

+34 90,6k 49 753

На работу на велосипеде. Еще одно мнение

+59 43,5k 117 324

О дисководах и их использовании на современных компьютерах

+48 24k 57 83