




Может войдёшь?

Черновики

Написать статью

Профиль

Представляем Lumen

Introducing Lumen →  Laracasts +1 0007 июня 2016 

перевод

Laracasts

Lumen

Это перевод видео-урока с [Laracasts](#) — [Introducing Lumen](#) от 14.04.2015.
Перевод обновлён 05.06.2016. **Опечатка?** Выдели и нажми Ctrl+Enter.

(0:00)

Привет, с вами Тейлор Отуэлл с ещё одним скринкастом по Laravel. В этом скринкасте я бы хотел познакомить вас с Lumen, новейшим членом семейства Laravel. Lumen – это официальный микро-фреймворк Laravel, и он молниеносно быстрый. Это отличный путь для написания микро-сервисов или небольших API, используя любимый вами синтаксис Laravel, но в очень быстром фреймворке.

(0:30)

Очень часто, когда у вас было основное приложение Laravel, но вы хотели поддержать его с микро-сервисами или другими вспомогательными сервисами, то вам приходилось брать Silex или Slim и пользоваться другим синтаксисом, а не уже привычными библиотеками.

И иногда, как вы знаете, это требует приличного объёма дополнительных усилий для настройки вашей БД, кэша и всего прочего.

С Lumen у вас появится вся мощь удобных компонентов Laravel, так что Eloquent, БД, кэширование (даже Redis-кэширование) – всё это доступно вам по умолчанию и при этом всё равно будет быстрее чем все другие варианты.

(1:00)

Так что это отличный выбор для поддержки ваших приложений Laravel. Позвольте я вам покажу как он выглядит.

У меня тут установленный по умолчанию Lumen и вы можете увидеть что структура папок тут потоньше. Нет директорий `resources` и `database` по умолчанию. Но есть знакомый `App/Http/routes.php`.

И вы увидите, что мы используем замыкания, хотя в Lumen вы можете пользоваться контроллерами с полным внедрением зависимости, следовательно эта мощь вам также доступна если захотите.

(1:30)

Переводы Laracasts

1. Установка Laravel для новичков
2. Представляем Lumen

Основы Laravel 5

1. Встречайте Composer
2. Виртуальные машины и Homestead
3. Введение в маршрутизацию, контроллеры и представления
4. Передача данных в шаблоны
5. Blade 101
6. Среды и настройки
7. Миграции
8. Eloquent 101
9. Основы Model/Controller/View
10. Формы
11. Даты, мутаторы и области запросов
12. Запросы из форм и проверка ввода
13. Частичные шаблоны и повторное использование форм
14. Отношения в Eloquent
15. Простая аутентификация
16. Посредники - почти что огры
17. Повторим изученное
18. Привязка моделей к маршрутам
19. Управление ресурсами приложения
20. Сообщения-вспышки
21. Многие-ко-многим (с тегами)
22. Выбор тегов в UI
23. Синхронизация тегов
24. Улучшаем элементы select
25. Косвенная передача данных в шаблоны
26. Сервис-контейнер

И конечно можно создавать посредников (middleware) для защиты ваших маршрутов и некоторой фильтрации до того как запрос попадает в ваше приложение. ОК, процесс создания маршрута в Lumen такой же, как и создания маршрута в Laravel. Мы можем просто вернуть тут `Hello World`, вот здесь. И вызовем это в нашем браузере. И получим: «Hello World». И конечно всё это очень быстро. И что ещё круто... давайте запросим немного данных у БД и я покажу вам как просто это сделать в Lumen.

(2:00)

Хорошо, итак, если я перейду в мой файл окружения – именно здесь я сконфигурирую приложение Lumen. Здесь нет директории `config` с 7-8 файлами, а есть лишь один конфигурационный файл и вы можете использовать библиотеку `dotenv`, которую использует Laravel, или можете загрузить свои переменные окружения через `nginx` или ещё каким-то образом с вашего производственного сервера. Хорошо, как видите, у меня уже есть БД сконфигурированная для Homestead. Давайте создадим таблицу. И что тут в Lumen ещё здорово – с ним идут некоторые команды Artisan.

(2:30)

Давайте я вам это покажу:

```
cd Laravel/lumen
```

Хорошо, теперь если я сделаю:

```
php artisan list
```

вы увидите, что у меня есть некоторые команды миграции, очереди, т.е. Lumen включает также очень лёгкий обработчик очередей, и даже может добавлять новые задачи в очередь самостоятельно, как и полноценный Laravel. Так как очереди – это большая часть современных веб-приложений, то для Lumen есть смысл уметь и ставить в очередь и использовать сообщения из очереди от себя.

Хорошо, давайте продолжим и создадим миграцию, чтобы для начала просто создать элементарную таблицу.

(3:00)

И сперва нам нужно создать папку для этой БД, потому что по умолчанию её там нет. Я делаю это так:

```
php artisan make:database
```

Хорошо, и вы увидите созданную папку БД с директориями `migrations` и `seeds`. То есть мы можем также и делать `seed` (начальную загрузку) для нашей БД. ОК, я вернусь в свой терминал, и теперь мы готовы создать миграцию, так что давайте создадим таблицу `users` и дадим ей такое имя:

```
php artisan make:migration create_users_table --create=users
```

(3:30)

Хорошо, эта миграция создана. Теперь что нам нужно сделать – это

27. [Подтягиваем хвосты и закругляемся](#)

Мастерство Vim

1. [Привет, Vim](#)
2. [Осматриваемся вокруг](#)
3. [Горячие клавиши и команды](#)
4. [A Prettier Vim](#)
5. [Разбиение окна](#)
6. [Vundle и улучшенный обзор файлов](#)
7. [Быстрый поиск с CtrlP](#)
8. [Подкручиваем настройки](#)
9. [Перемещаемся под код с Stags](#)
10. [Настройка подсветки синтаксиса](#)
11. [Поиск и замена по всему проекту](#)
12. [PeerOpen](#)
13. [Настройки для Laravel](#)
14. [Управление сниппетами](#)
15. [Обрамление текста](#)
16. [Оптимизации для PHP](#)
17. [Автодополнение](#)
18. [Automatic PSR-2 Formatting](#)
19. [Метки](#)
20. [Табуляция, отступы и пробелы](#)
21. [Развлекаемся с макросами](#)
22. [PHP Documentor и Ultisnips](#)



Laracasts +1 000

Сайт: laravel.ru

Статистика: Символов —
7 174/5 997 без пробелов
(6 374/5 329 без кода); слов —
1 046

Разметка: Wiki

Исходник статьи

добавить её к настройкам автозагрузки. Так что добавим `classmap` и просто напишем тут `database/`:

```
"classmap": [  
    "database/"  
]
```

Хорошо, теперь нам нужно заново создать файлы автозагрузки:

```
so
```

ОК, теперь мы готовы к созданию этой миграции. Давайте перейдём внутрь файла и добавим здесь пару полей.

(4:00)

Например e-mail адрес, и конечно наши временные штампы. Для этого демо – это всё, что нам нужно:

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

Хорошо, мы почти готовы к запуску этой миграции.

Последнее что осталось сделать – раскомментировать несколько строк в нашем файле `bootstrap/app.php`. Если я открою его, мне нужно раскомментировать вот эту вызов `Dotenv::load()`. Она на самом деле загружает наши переменные окружения в само окружение. И конечно на вашем производственном сервере вы можете делать это как угодно, через `nginx` или любые другие способы загрузки переменных через окружения. Также, мне нужно раскомментировать вот это:

```
$app->withFacades();
```

(4:30)

Это позволит нашей миграции использовать этот фасад `Schema` для создания нашей таблицы. Хорошо, убрав комментарии с этих двух строк, моя миграция должна быть готова к запуску. Вернёмся в терминал. Мне нужно быть в моей виртуальной машине, так как моя БД живёт там в `Homestead`, и я готов к запуску:

```
php artisan migrate --force
```

(мы форсируем запуск)

Хорошо, наша таблица `users` была создана, и это здорово!

Отлично, теперь давайте используем `Eloquent` чтобы наполнить БД какими-то данными.

(5:00)

Итак, первое что мне требуется сделать, это именно включить `Eloquent`, так как по умолчанию он не активирован. Я могу сделать это снова в моём файле `bootstrap/app.php` – нужно лишь раскомментировать эту строку:

```
$app->withEloquent();
```

И это подгрузит `Eloquent` при загрузке моего приложения. Хорошо, теперь перейдём в мой файл с маршрутами, и прямо здесь создадим простую модель для этого демо. Создадим модель `User`

и она унаследует `Illuminate\Database\Eloquent\Model`:

```
class User extends Model {  
}
```

Хорошо, и нам не нужно задавать имя таблицы, поскольку Eloquent сам за нас это решит.

(5:30)

Теперь давайте создадим нового пользователя и зададим адрес почты как `taylor@laravel.com`, сохраним этого пользователя и вернём `All Done!`. Итак:

```
$user = new User;  
$user->email = 'taylor@laravel.com';  
$user->save();  
return 'All Done!';
```

И теперь давайте вызовем этот маршрут и посмотрим, создан ли наш пользователь.

```
lumen.app:8000
```

Хорошо, и мы получили: «All Done!», так что давайте проверим БД. Если я обновлю это, вы увидите – у нас тут есть таблица `users` и вон он наш пользователь в БД.

(6:00)

Итак, мы уже используем мощь Eloquent с Lumen. Пойдём дальше и вернём данные Eloquent из этого маршрута. Хорошо, мы смотрим на нашу БД – есть пользователь с ID = 1, так давайте сделаем ещё один маршрут с ID пользователя `user/{id}`, так чтобы мы тут могли принимать параметр. Мы сделаем:

```
$app->get('user/{id}', function ($id) {  
    return User::findOrFail($id);  
});
```

(6:30)

Очень простой маршрут, который берёт пользователя из БД и выдаёт его обратно на страницу. Вызовем этот маршрут:

```
lumen.app:8000/user/1
```

Хорошо, пользователь 1, и как вы видите, как и в Laravel, мы легко получили JSON пользователя. Нет ничего проще, если вы хотите начать строить API с Lumen и Eloquent, и вы можете даже с лёгкостью добавить сюда кэширование. ОК, давайте вызовем пользователя, которого нет в БД:



```
lumen.app:8000/user/10
```

Как видите, мы должны получить исключение 404, и вот оно – отчёт с ошибкой в стиле Symfony, когда что-то в Lumen идёт не так.

(7:00)

С помощью этого вы легко разберётесь с тем, что произошло в

вашем приложении. Вот насколько просто начать создавать на Lumen и Eloquent, и конечно мы можем использовать кэширование и даже очереди задач с мощностью этого микро-фреймворка с поразительной скоростью работы.

Как вы считаете, полезен ли этот материал?  Да  Нет

 +6 

 7 136

 8K

Двигается на Habravel

Написать комментарий

Разметка:

 Wiki ?  Markdown ?

Какие мысли на этот счёт?

Авторизуйся, чтобы прокомментировать.