Фреймворк Laravel

Самостійна робота до курсу “Програмування Інтернет”, Запорізька державна інженерна академія (ЗДІА), доцент Попівщий В.І., 2016 р.

Зміст

- Огляд Laravel
- Як інсталювати Laravel 5.2 в XAMPP (Windows)
- Як розробляти сайт в Laravel

Передумови (для Laravel 5.2)

- Комп’ютер, з’єднаний з Інтернет
- XAMPP 5.6.28 (https://www.apachefriends.org/index.html) в якості локального сервера (вимоги для Laravel 5.2:
  - PHP >= 5.5.9
  - OpenSSL PHP Extension
  - PDO PHP Extension
  - PHP Mbstring Extension
  - Tokenizer PHP Extension)

  Весь цей набір компонентів присутній в XAMPP та OpenServer 5.2.5 (http://open-server.ru/) під Windows

Огляд Laravel

- **Laravel - безкоштовний веб-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC**
- Eloquent ORM - реалізація шаблону проектування ActiveRecord на PHP
- Зворотня маршрутизація
- REST-контролери
- Автозавантаження класів (без include)
- Міграції - система управління версіями для баз даних
- Модульне тестування (юніт-тести)
- Сторінкове виведення(Pagination) - спрощує генерацію сторінок

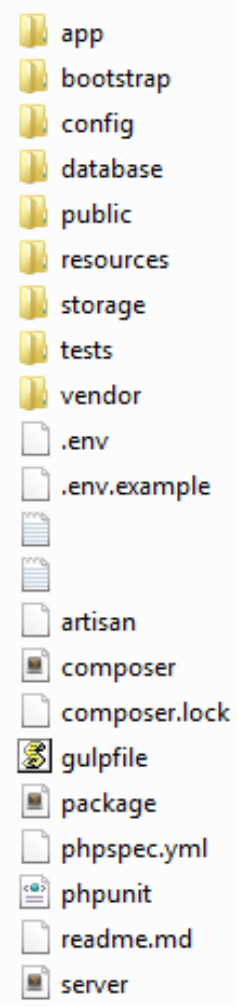
Історія Laravel

- Laravel стартував у 2011 році.
- Ключовий розробник: Тейлор Отвел (Taylor Otwell).
- Нагороди: У 2015 році в результаті опитування sitepoint.com по використанню PHP-фреймворків серед програмістів **зайняв перше місце** в номінаціях: Фреймворк корпоративного рівня, Фреймворк для особистих проектів.
- Сайт https://laravel.com/
- Підтримка: https://laravel.ru/, http://laravel.su/,
- Поточна версія (на листопад 2016 р.) – Laravel 5.3

Інсталяція Laravel 5.2 на XAMMP

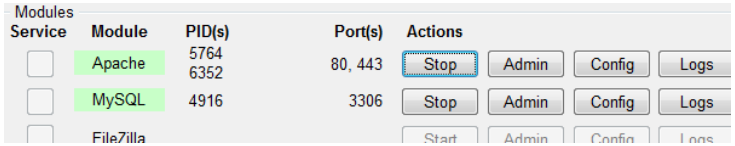
- **КРОК 1**. Встановлення локального сервера (XAMPP 5.6.28)
- **КРОК 2**. Встановлення пакетного менеджера Composer (https://getcomposer.org/Composer-Setup.exe).
  - В ході інсталяції вказати місцезнаходження файлу php.exe
  - Якщо інсталювати на Обчислювальних Центрах ЗДІА, то указати адресу proxy-сервера (http://192.168.1.1:3128)
- **КРОК 3**. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог **xampp/htdocs** і подайте команду:
  composer create-project laravel/laravel mysite "5.2.*"

  (після закінчення буде створено каталог xammp/htdocs/mysite з наступним вмістом)



Інсталяція Laravel 5.2 на XAMMP

- **КРОК 4**. Активізуйте Apache і MySql з панелі управління XAMPP:



- **КРОК 5**. Наберіть в браузері: **localhost/mysite/public** (повинен з’явитись екран Laravel 5)

Наступні кроки 6,7,8,9 бажано виконати після налаштування бази даних застосунку (дивись далі).

- КРОК 6. Для подальшої зручності в роботі бажано сконфігурувати віртуальний хост в XAMMP. Для цього треба відкрити в редакторі (наприклад Notepad++) файл **C:\xampp\apache\conf\extra\httpd-vhosts.conf** (якщо у вас XAMMP на диску C:\) і додати наступні рядки в кінець файлу:

Інсталяція Laravel 5.2 на XAMMP

```
# VirtualHost for LARAVEL.DEV
<VirtualHost  *:80>
        ServerAdmin webmaster@dummy-host2.laravel.dev
        DocumentRoot "C:/xampp/htdocs/mysite/public“
        ServerName  laravel.dev
        ErrorLog  “logs/laravel.dev-error.log”
        CustomLog  “logs/laravel.dev-access.log” common
</VirtualHost>
```

Після цього наш Apache буде прослуховувати **laravel.dev** з’єднання

Інсталяція Laravel 5.2 на XAMMP

- КРОК 7. Залишилось внести зміни в файл **C:\Windows\System32\drivers\etc\hosts**

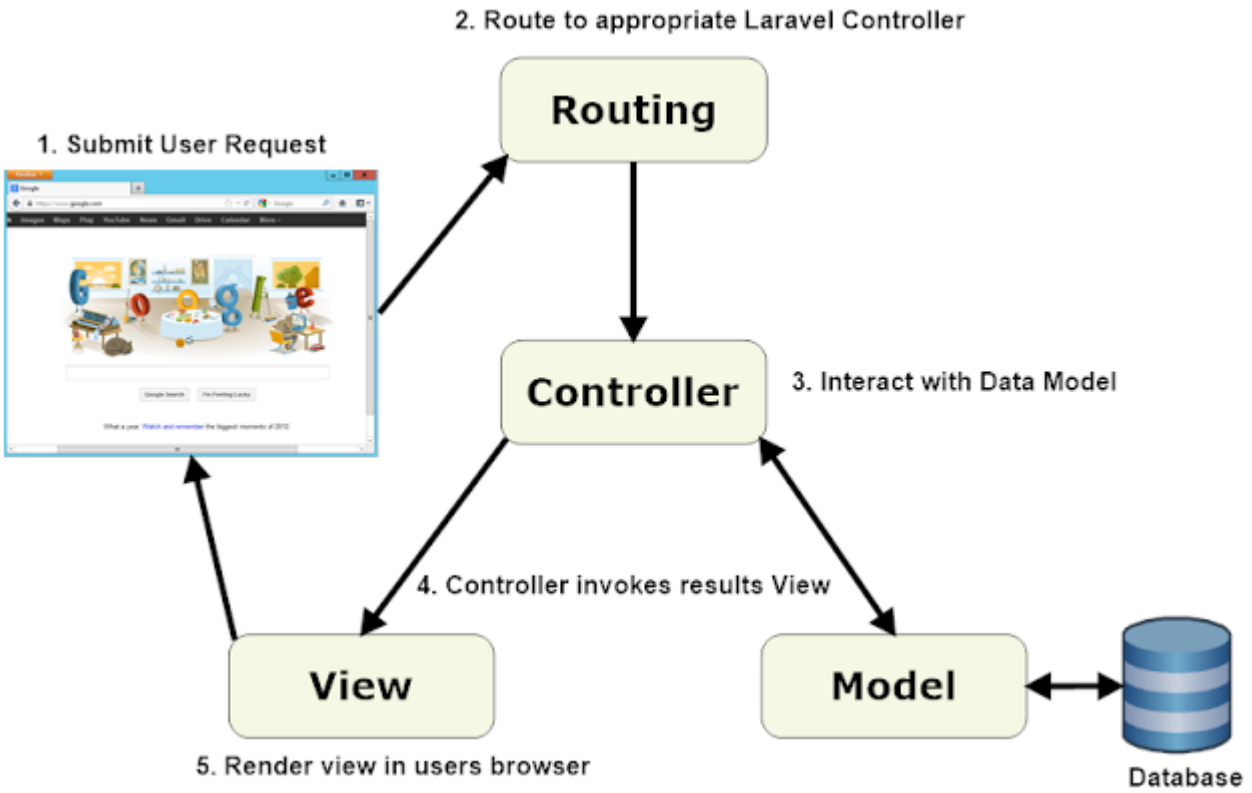
  Додайте в файл **hosts** рядок:

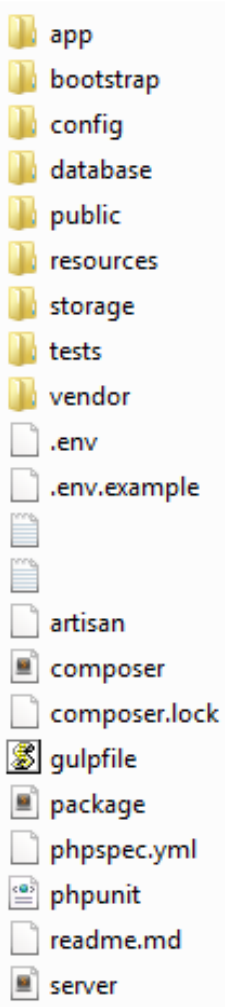
  127.0.0.1 laravel.dev

- КРОК 8. (тільки на Обчислювальних Центрах ЗДІА)
  Включіть адресу **laravel.dev** в виключення проксі-сервера в налаштуваннях вашого браузера.
- КРОК 9. Наберіть в браузері: **laravel.dev**
  (повинен з’явитись екран Laravel 5)

Файлова структура фреймворку

- Laravel використовує архітектуру MVC.





Налаштування фреймворку

- Відкрийте файл .env і відредагуйте деякі рядки:

  DB_CONNECTION=mysql

  DB_HOST=127.0.0.1

  DB_PORT=3306

  DB_DATABASE=mylaravel

  DB_USERNAME=root

  DB_PASSWORD=

- Увійдіть в phpMyadmin (http://localhost/phpmyadmin) і створіть базу даних **mylaravel** (з utf8_general_ci)

Побудова простого застосунку (1)

- Для демонстрації можливостей Laravel побудуємо застосунок, що працює з машинами.

Модель Car

- Laravel має вбудований в інтерфейс командного рядка ремісник (Artisan), який надає вам багато корисних команд для побудови додатка.
- **КРОК 1**. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог застосунку (**xampp/htdocs/mysite**) і подайте команду для генерації нової моделі Car:
  php artisan make:model Car --migration
- Всі моделі зберігаються в каталозі **app**, так що попередня команда згенерує файл **app/Car.php** з наступним кодом:

Побудова простого застосунку (2)

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Car extends Model
{
    //
}
```

- Завдяки вбудованій функціональності моделі Laravel, створивши порожній клас моделі, Laravel буде вважати, що ця модель пов'язана з таблицею бази даних з ім'ям **cars**.
- Завдяки прапорцю **--migration** при створенні моделі, Laravel також генерує міграційний файл бази даних для створення таблиці бази даних (https://laravel.com/docs/5.2/migrations). Файл міграції **database/migrations/[мітка часу] _create_cars_table.php** містить наступний код:

Побудова простого застосунку (3)

. . . . . .

```
public function up() {
Schema::create('cars', function (Blueprint $table) {
    $table->increments('id');
    $table->timestamps();
}); }
```

. . . . . .

КРОК 2. Додайте три додаткових стовпців, які будуть зберігати характеристики автомобілів (виробника, модель та дату виробництва):

```
$table->string('make');
$table->string('model');
$table->date('produced_on');
```

КРОК 3. Подайте Artisan-команду для запуску міграції, що створить таблицю **cars**:

php artisan migrate

Побудова простого застосунку (4)

- **КРОК 4**. Додайте вручну декілька записів в таблицю автомобілів (**cars**) бази даних нашого застосунку (**mylaravel**). Для цього наберіть в браузері

  localhost/phpmyadmin і т.д.

Контролер

- В Laravel, тип об'єкта - такий, як Car, в даному випадку - називають в якості ресурсу (**resource**).
- **КРОК 5**. Створіть контролер ресурсів (**resource controller**) - контролер для обробки всіх запитів, пов'язаних з ресурсом Car - за допомогою іншої простої команди Artisan:

  php artisan make:controller CarController

  Це створить файл **app/Http/Controllers/CarController.php** контролера з відповідним кодом.

Побудова простого застосунку (5)

Маршрути (Routes)

- На попередньому кроці Laravel автоматично згенерував контролер з усіма типовими CRUD операціями.
- Тепер нам просто потрібно визначити маршрути, щоб зв'язати URL-адреси з усіма цими діями контролера.
- Знову ж таки, оскільки це такий загальний сценарій, ви можете визначити єдиний маршрут (**resource route**), який створює маршрути для всіх дій контролера ресурсів.
- **КРОК 6**. У файлі конфігурації маршрутів – **app/Http/routes.php** —додайте наступне визначення маршруту для ресурсу Car:

  Route::resource('cars', 'CarController');

- Це єдине визначення маршруту буде визначати всі маршрути, пов'язані з нашим ресурсом Car :

Побудова простого застосунку (6)

Request Type	Path	Action	Route Name
GET	/cars	index	cars.index
GET	/cars/create	create	cars.create
POST	/cars	store	cars.store
GET	/cars/{car}	show	cars.show
GET	/cars/{car}/edit	edit	cars.edit
PUT/PATCH	/cars/{car}	update	cars.update
DELETE	/cars/{car}	destroy	cars.destroy

- Тепер, наприклад, давайте здійснимо реалізацію сторінки Show Car (машини з конкретним id).

Побудова простого застосунку (6)

Метод дії show контролера

- Згідно з попередньою таблицею, сторінка Show Car буде доступною за адресою: http://app.url/cars/ {car} (в нашому випадку app.url – це **laravel.dev**). В цьому випадку {car} буде, насправді, ідентифікатором id об'єкта автомобіля в базі даних.
- Так, наприклад, URL, щоб переглянути автомобіль з ідентифікатором 1 буде http://app.url/cars/1
- З метою реалізації сторінки Show Car, в show дії контролера, нам необхідно:
  - Використати модель Car, щоб отримати відповідний об'єкт автомобіля з бази даних.
  - Завантажити вид (view) для сторінки Show Car, і передати йому об'єкт Car, витягнутий з бази даних.

Побудова простого застосунку (7)

- **КРОК 7**. По-перше, для того, щоб отримати доступ до моделі Car в контролері, нам потрібно додати новий вираз вище класу контролера (файл **app/Http/Controllers/CarController.php**):

```
use App\Car;
```

- **КРОК 8.**Потім ми можемо завершити метод дії **show** наступним кодом:

```
public function show($id)
{
    $car = Car::find($id);
    return view('cars.show', array('car' => $car));
}
```

Побудова простого застосунку (8)

Вид (View)

- Файли видів Laravel всі зберігаються в папці **resources/views**. І вони можуть бути організовані в підпапках в цьому каталозі.
- У попередньому кроці, ми послались на вид з назвою **cars.show**. Це говорить Laravel шукати файл виду, розташований в підпапці **cars** (всередині головного каталогу видів **resources/views**) під назвою **show.blade.php**.
- Файли видів Laravel використовують шаблонний движок **Blade**, і, отже, називаються *.blade.php
- **КРОК 9.** Створіть каталог **cars** всередині каталогу **resources/views**

Побудова простого застосунку (9)

- **КРОК 10.** Створіть файл **resources/views/cars/show.blade.php** з наступним кодом:

```
<!DOCTYPE html>
<html>
<head> <title>Car {{ $car->id }}</title> </head>
<body>
<h1>Car {{ $car->id }}</h1>
    <ul>
        <li>Make: {{ $car->make }}</li>
        <li>Model: {{ $car->model }}</li>
        <li>Produced on: {{ $car->produced_on }}</li>
    </ul>
</body>
</html>
```

Побудова простого застосунку (10)

Метод дії index контролера

- В цьому методі дії необхідно вивести всі автомобілі
- **КРОК 11.** В файл **app/Http/Controllers/CarController.php** додайте метод дії **index** з наступним вмістом:

```
public function index()
{
    $cars = Car::orderBy(‘id’, ‘desc’)->paginate();
    return view('cars.index', compact('cars'));
}
```

Побудова простого застосунку (11)

Вид для перегляду всіх автомобілів

- **КРОК 12.** Створіть файл **resources/views/cars/index.blade.php** з наступним кодом:

```
@foreach($cars as $car)
<car>
    <h1>Car {{ $car->id }}</h1>
   <ul>
     <li>Make: {{ $car->make }}</li>
     <li>Model: {{ $car->model }}</li>
     <li>Produced on: {{ $car->produced_on }}</li>
   </ul>
</car>
<hr>
@endforeach
```

Тестування застосунку

- Виконайте кроки 6 і 7 по конфігуруванню віртуального хоста **laravel.dev** (слайди 7, 8, 9 даної презентації)
- Наберіть в браузері: **http://laravel.dev/cars/**
(повинен з’явитись список всіх машин, що внесені в базу даних)
- Наберіть в браузері: **http://laravel.dev/cars/1**
(повинна з’явитись інформація про першу машину)

Подальше вдосконалення застосунку

- Вивчіть документацію, потрібну для додавання в застосунок можливості додавання нових машин, редагування та видалення інформації.

Демо сайт з авторизацією (1)

- **КРОК 1**. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог **xampp/htdocs** і подайте команду:
  composer create-project laravel/laravel larasite "5.2.*"

  (після закінчення буде створено каталог xammp/htdocs/larasite)
- **КРОК 2**. Активізуйте Apache і MySql з панелі управління XAMPP
- **КРОК 3**. Наберіть в браузері localhost/phpmyadmin та створіть базу даних **larasite** (з utf8_unicode_ci)
- **КРОК 4**. Перейдіть в каталог застосунку (**xampp/htdocs/larasite**) і відредагуйте деякі рядки файлу .env:

  DB_DATABASE=larasite
  DB_USERNAME=root
  DB_PASSWORD=

Демо сайт з авторизацією (2)

- **КРОК 5**. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог застосунку (**xampp/htdocs/larasite**) і подайте команду:

  php artisan make:auth

- В результаті будуть автоматично згенеровані всі необхідні **views** та контролер для авторизації.
- **КРОК 6**. Для створення таблиці **users** нашої бази даних подайте команду:

  php artisan migrate

- **КРОК 7**. Для