Indice

Laravel

Laravel es el framework de código abierto más popular de 2015-2016 para desarrollar aplicaciones y servicios web con PHP 5 y PHP 7.

Su filosofía es desarrollar código PHP de forma elegante y simple.

¿Por qué Laravel?

Laravel se caracteriza por su popularidad, sencillez y apoyo a las buenas prácticas de php.

Este framework en GitHub hasta ahora tiene 32.737 estrellas y tiene una fuerte presencia en la industria.

Una de las principales razones que hace que Laravel sea tan popular es que su objetivo es hacer feliz al desarrollador, esto se ve reflejado en las facilidades que nos otorga como:

- Flexibilidad a la hora de implementar la lógica
- Generadores de componentes desde la línea de comandos
- Facilidad y utilidades a la hora de testear el código
- Laravel-mix es un paquete para facilitar los css y javascript necesarios para nuestra aplicación
- Sistema de autenticación completo para un prototipo rápido
- Forma sencilla de validar datos de entrada de nuestra aplicación

Otra razón de su grandeza es la cantidad de documentación y recursos disponibles en la web que hay para este framework.

A continuación podemos observar una comparativa de Laravel con otros frameworks destacados de PHP:

	Laravel	**Symfony**	**Codeigniter**	**Zend**
Lanzamiento	Junio 2011	Octubre 2005	Febrero 2006	Junio 2005
Estrellas GitHub	32.737	14.680	14.575	5.522
Última versión	5.4.18 2 de marzo 2017	3.2.5 9 de marzo 2017	3.1.4 20 marzo 2017	3.0.0 28 junio 2016
Contribuidores	406	1.454	416	690
Commits	5.370	31.583	9.548	27.056

Composer

El primer paso antes de empezar a usar Laravel es instalar Composer, utilizado para las dependencias. Composer es un administrador de dependencias para PHP.

Su manera de funcionar es extrayendo todas las librerías, dependencias necesarias y las administrándolas todas en un solo lugar.

Este tipo de gestión de dependencias en un proyecto no es un concepto nuevo, y de hecho, gran parte de Composer está realmente inspirado en npm de Node.js y Bundler de Ruby.

Instalar Composer

Instalar Composer es realmente fácil. Desde línea de comandos:

```
$ sudo mv composer.phar /usr/local/bin/composer
$ composer
```

Instalar en Windows

Descarga e instala la última versión de Composer y configura el PATH para que composer pueda ejecutarse desde cualquier directorio en la línea de comandos.

Composer-Setup.exe

Ejemplo de uso

If you want to use Laravel framework, on your *composer.json* file:

```
{
  "require": {
    "laravel/framework": "5.4.*"
  }
}
```

Next run on your command line:

```
$ composer install
```

Blade

Blade es un sistema de plantillas sencillo pero potente provisto de Laravel. Blade es impulsado por herencias y secciones de plantillas. Todas las plantillas de Blade deben utilizar la extensión .blade.php.

La plantilla de Blade se interpreta a PHP con lo que cualquier código PHP es válido para Blade.

Nos ofrece una forma sencilla de protegernos de ataques XSS al escapar todas las variables por defecto.

Es sencillo añadir directivas propias.

Ejemplo de uso

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

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

@section('sidebar')
    @parent

    <p>This is appended to the master sidebar.</p>
@stop

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

Añadir directivas

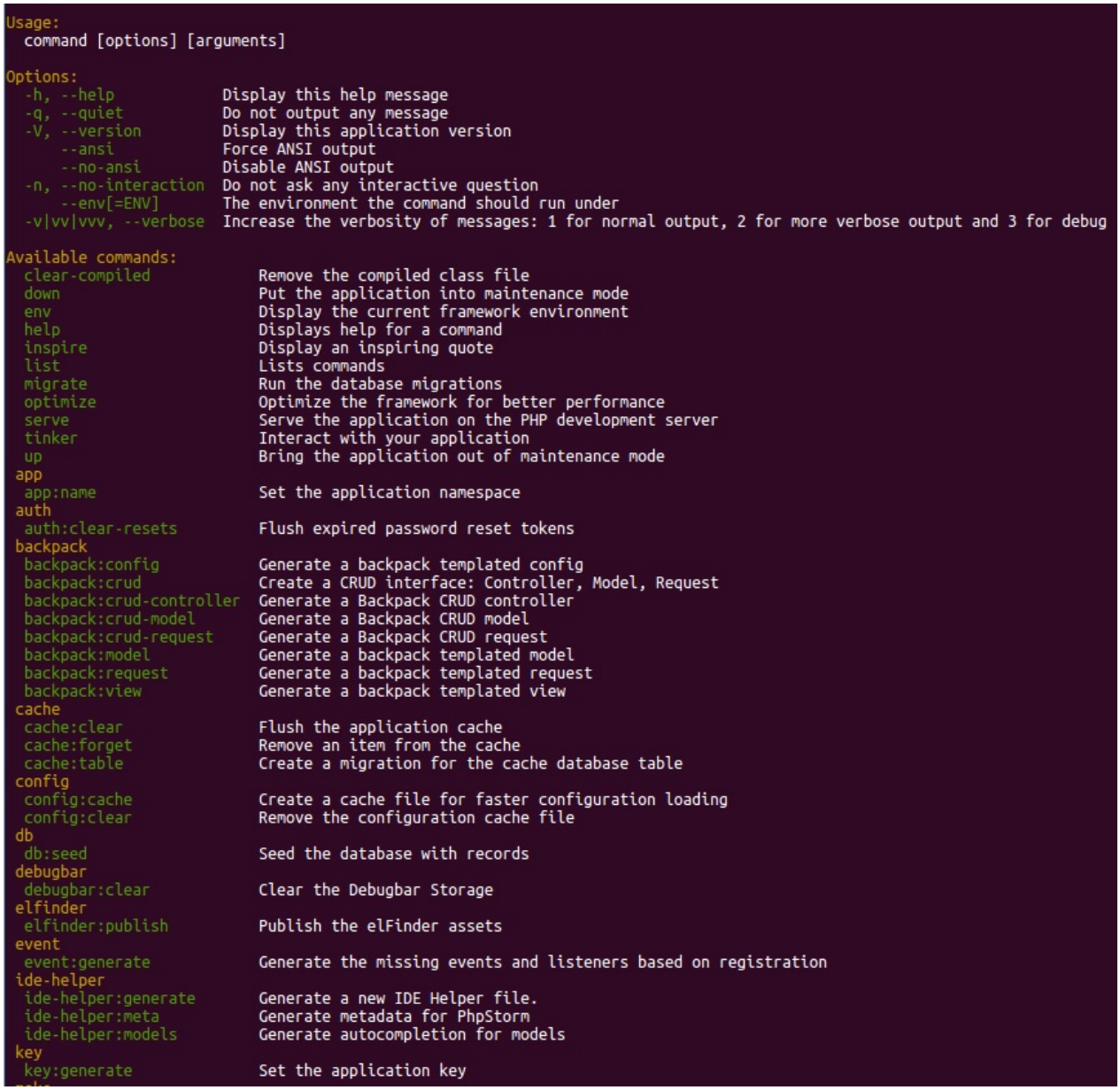
```
protected function registerBladeExtensions()
    {
        $this->app->afterResolving('blade.compiler', function (BladeCompiler $bladeCom
piler) {
            $bladeCompiler->directive('role', function ($arguments) {
                list($role, $guard) = explode(',', $arguments.',');
                return "<?php if(auth({$guard})->check() && auth({$guard})->user()->ha
sRole({$role})): ?>";
            });
            $bladeCompiler->directive('endrole', function () {
                return '<?php endif; ?>';
            });
            $bladeCompiler->directive('hasrole', function ($arguments) {
                list($role, $guard) = explode(',', $arguments.',');
                return "<?php if(auth({$guard})->check() && auth({$guard})->user()->ha
sRole({$role})): ?>";
            });
            $bladeCompiler->directive('endhasrole', function () {
                return '<?php endif; ?>';
            });
            $bladeCompiler->directive('hasanyrole', function ($arguments) {
                list($roles, $guard) = explode(',', $arguments.',');
                return "<?php if(auth({$guard})->check() && auth({$guard})->user()->ha
sAnyRole({$roles})): ?>";
            });
            $bladeCompiler->directive('endhasanyrole', function () {
                return '<?php endif; ?>';
            });
            $bladeCompiler->directive('hasallroles', function ($arguments) {
                list($roles, $guard) = explode(',', $arguments.',');
                return "<?php if(auth({$guard})->check() && auth({$guard})->user()->ha
sAllRoles({$roles})): ?>";
            });
            $bladeCompiler->directive('endhasallroles', function () {
                return '<?php endif; ?>';
            });
        });
    }
```

Artisan

Artisan es la interfaz de línea de comandos utiizada en Laravel.

Proporciona una serie de comandos útiles de gran ayuda mientras construye su aplicación. Para ver una lista de todos los comandos de Artisan disponibles, puede utilizar el comando:

```
php artisan list
```



```
make
 make:auth                  Scaffold basic login and registration views and routes
 make:command               Create a new Artisan command
 make:controller            Create a new controller class
 make:event                 Create a new event class
 make:job                   Create a new job class
 make:listener              Create a new event listener class
 make:mail                  Create a new email class
 make:middleware            Create a new middleware class
 make:migration             Create a new migration file
 make:migration:pivot       Create a new migration pivot class
 make:migration:schema      Create a new migration class and apply schema at the same time
 make:model                 Create a new Eloquent model class
 make:notification          Create a new notification class
 make:policy                Create a new policy class
 make:provider              Create a new service provider class
 make:request               Create a new form request class
 make:seed                  Create a new database seed class
 make:seeder                Create a new seeder class
 make:test                  Create a new test class
migrate
 migrate:fresh              Drop all tables from db and rebuild it using migrations.
 migrate:install            Create the migration repository
 migrate:refresh            Reset and re-run all migrations
 migrate:reset              Rollback all database migrations
 migrate:rollback           Rollback the last database migration
 migrate:status             Show the status of each migration
notifications
 notifications:table        Create a migration for the notifications table
queue
 queue:failed               List all of the failed queue jobs
 queue:failed-table         Create a migration for the failed queue jobs database table
 queue:flush                Flush all of the failed queue jobs
 queue:forget               Delete a failed queue job
 queue:listen               Listen to a given queue
 queue:restart              Restart queue worker daemons after their current job
 queue:retry                Retry a failed queue job
 queue:table                Create a migration for the queue jobs database table
 queue:work                 Start processing jobs on the queue as a daemon
route
 route:cache                Create a route cache file for faster route registration
 route:clear                Remove the route cache file
 route:list                 List all registered routes
schedule
 schedule:run               Run the scheduled commands
session
 session:table              Create a migration for the session database table
storage
 storage:link               Create a symbolic link from "public/storage" to "storage/app/public"
vendor
 vendor:publish             Publish any publishable assets from vendor packages
view
 view:clear                 Clear all compiled view files
```

Con `php artisan serve` se corre la aplicación en el servidor local de laravel.

Una de las grandes utilidades de **artisan** son los generadores: es posible desde línea de comandos, con artisan crear modelos, controladores, middlewares... Todo esto está en la sección ***make:***

Tinker

Todas las aplicaciones de Laravel incluyen Tinker. Te permite interactuar con toda la aplicación de Laravel en la línea de comandos, incluyendo el ORM de Eloquent, trabajos, eventos y mucho más. Para entrar en el entorno de Tinker, hay que ejecutar el comando Artisan de tinker:

```
php artisan tinker
```

Por ejemplo si quieres ver rápidamente una lista con todos los usuarios de tu aplicación:

```
php artisan tinker

App\Models\User::all()
```

Para ver las encuestas pertenecientes al usuario con id 1:

Inyección de dependencias

Es un patrón de diseño extremadamente sencillo. La inyección de dependencias permite pasar a través del constructor de la clase todos los objetos que necesita una clase para funcionar.

Por ejemplo:

Un móvil depende de una batería para funcionar. Un código sin inyección de dependencias sería así:

```
<?php
class Mobile {
    public function __construct() {
        $batery = new Batery();
        $batery->isUsable();
    }
}
```

El problema con la implementación anterior, es que la clase “Batery” se declara implícitamente, no es posible saber que existe dicho objeto hasta inspeccionar código fuente de la clase Mobile. Por lo tanto no solamente la declaración es implícita sino que las clases también están acopladas.

Es decir, no puedes sustituir la clase Batery, por Batery2900mAh o Batery2100mAh sin modificar el código de la clase Mobile. De esta misma manera, la clase se vuelve difícil de mantener y difícil de depurar. Es aquí donde entra en juego la inyección de dependencia, ya que usando este patrón estás haciendo explicita la declaración de la clase y por lo tanto desacoplando Mobile de Batery. El código anterior, cambia a:

```
<?php

interface Mobile {
    pubic function isUsable();
}

class Batery2900mAh implements Mobile {

    public function isUsable()
    {
        echo "charge me!";
    }
}

class Batery2100mAh implements Mobile {

    public function isUsable()
    {
        echo "batery charged";
    }
}

class MobileOwner {
    protected $mobile;

    public function __constructor(Mobile $mobile)
    {
        $this->mobile = $mobile;
    }

    public function using()
    {
        echo "Im using my mobile phone";
        $this->mobile->using();
    }
}
```

En los service providers iría el siguiente código, lo que hace posible pasando la interfaz Animal para registrar un nuevo gato por ejemplo.

```
$this->app->bind('Mobile', function ($app) {
    return new Batery2100mAh;
});
```

Paquetes

Developer Tools

- IDE Helper - Generates a helper file for IDE auto-completion
- Laravel 5 Extended Generators - Extends built-in file generators

Debugging & Profiling

- Debug Bar - Integrates PHP Debug Bar with Laravel
- Laravel 5 Log Viewer - Log viewer

Authentication & Authorisation

- JWT Auth- JSON Web Token authentication for APIs
- Laravel Permission - Associate users with roles and permissions
- Socialite - OAuth authentication with Facebook, Google, Twitter etc
- Socialite Providers 2.0 - 100+ social authentication providers for Socialite with Lumen support
- Google2FA - Google Two-Factor Authentication Module

Utilities

- Charts - Multi-library chart package to create interactive charts
- Eloquent Filter - An Eloquent Way To Filter Laravel Models And Their Relationships
- Eloquent Sluggable - Create slugs for Eloquent models
- Eloquent Sortable - Sortable behaviour for Eloquent models
- Laravel Datatables - jQuery DataTables API for Laravel 4|5
- Laravel Dot Env Generator - Generate .env.gen file based on the project source code
- Laravel Excel - Import and export Excel and CSV files
- noCAPTCHA - Helper for Google's new noCAPTCHA (reCAPTCHA)

Working with Javascript

- Laroute - Generate Laravel route URLs from JavaScript

- PHP Vars to JavaScript Transformer - Pass server-side string/array/collection/whatever to JavaScript
- Javascript Validation - Use validation rules, messages, FormRequest and validators to validate forms in client side without need to write any Javascript code

Databases, ORMs, Migrations & Seeding

- Backup Manager - Backup and restore databases from S3, Dropbox, SFTP etc.
- iSeed - Generate a new seed file from an existing database table
- Laravel Repository - Repositories to abstract the database layer for Laravel 5

Search

- Laravel Search - Unified API for Elasticsearch, Algolia, and ZendSearch

APIs

- Laravel CORS - Add CORS (Cross-Origin Resource Sharing) headers support
- Laravel Fractal - Output complex, flexible, AJAX/RESTful data structures with Fractal in Laravel and Lumen

Payments

- Cashier - Subscription billing with Stripe
- Omnipay for Laravel - Integrate the Omnipay PHP library

Localization

- Laravel Translatable - Making Eloquent models translatable by storing translations as JSON

Third-party Service Integration

- Laravel Analytics - An opinionated Laravel 5 package to retrieve pageviews and other data from Google Analytics
- Laravel Facebook - Facebook API bridge
- Laravel GitHub - PHP GitHub API bridge

Comunidad y Documentación

Laravel tiene una gran cantidad de documentación y recursos en la web. Cuenta además con una serie de videotutoriales entretenidos que van añadiendo nuevo contenido cada semana (Laracasts).

Estos son algunas de las utilidades de Laravel en la web:

Documentación

- Laravel Website
- Laravel Documentation
- Laravel Forum
- Laravel Recipes

Videotutoriales

- Laracasts

Canales de Youtube

- Laracon
- Laracon EU
- php[architect]
- phpacademy

Comunidad

- Twitter (@laravelphp)
- Slack (#laravel)
- Reddit (/r/laravel)
- News
- Laravel.io Forums
- Laracasts Forums
- Google+ Community
- Laravel Hunt
- Meetup

Despliegue

Laravel tiene requerimientos para funcionar. Todos éstos vienen dados por Homestead.

Homestead es una máquina virtual que debería usarse siempre como entorno de desarrollo en local.

Si no usas Homestead, has de asegurarte de cumplir con los siguientes requerimientos:

- PHP >= 5.6.4
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension

Para desplegar una aplicación de Laravel en entorno de producción, una opción bastante recomendada sería Forge.

Forge es una herramienta de Laravel que ayuda a los desarrolladores a enfocarse en el desarrollo, aparte del hosting. Hace muy sencillo el despliegue en producción.

Forge se encarga de cosas como:

- Creación y provisión de un nuevo servidor
- Configuración de claves SSH
- Creación de dominios y subdominios
- Deploying desde GitHub
- Guardar scripts de la bash comunes para volver a ejecutarlos en muchos servidores
- Configurar los trabajos cron programados
- Firewall y seguridad preconfigurados
- Configurar New Relic y otros servicios
- Instalar SSL Certs

Bibliografía

http://www.laravelbestpractices.com/

http://www.valuecoders.com/blog/technology-and-apps/laravel-best-php-framework-2017/

https://laravel.com/docs/5.4

https://alanchavez.com/como-usar-inyeccion-de-dependencias-en-php/