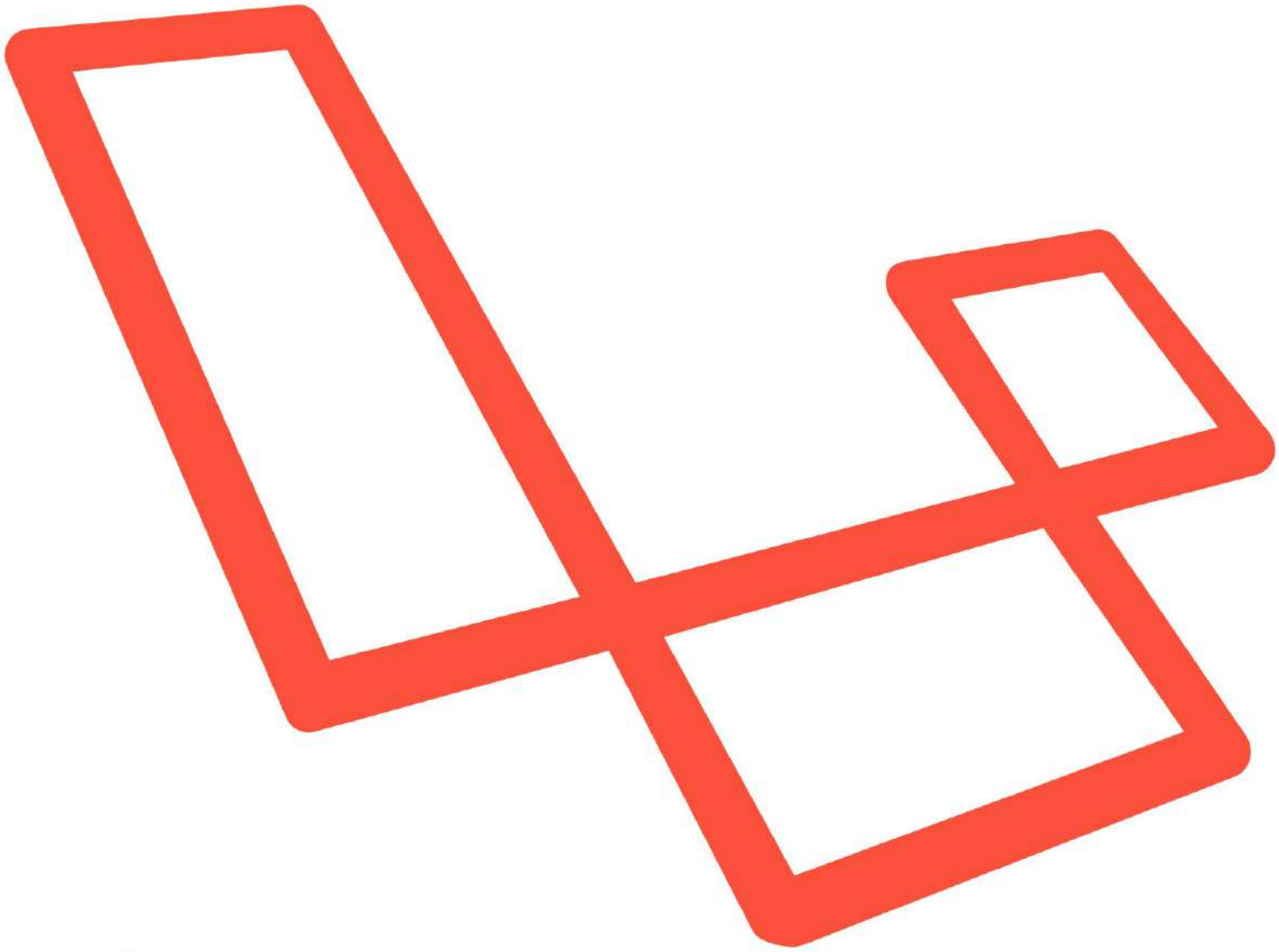

Libro laravel 5

Conceptos básicos y ejemplos

Table of Contents

Introducción a Laravel 5

Laravel es un framework para aplicaciones web con sintaxis expresiva y elegante. Creemos que el desarrollo debe ser una experiencia agradable y creativa para que sea verdaderamente enriquecedora. Laravel busca eliminar el sufrimiento del desarrollo facilitando las tareas comunes utilizadas en la mayoría de los proyectos web, como la autenticación, enrutamiendo, sesiones y almacenamiento en caché.

Laravel es un framework para el lenguaje de programación PHP. Aunque PHP es conocido por tener una sintaxis poco deseable, es fácil de usar, fácil de desplegar y se le puede encontrar en muchos de los sitios web modernos que usas día a día. Laravel no solo ofrece atajos útiles, herramientas y componentes para ayudarte a conseguir el éxito en tus proyectos basados en web, si no que también intenta arreglar alguna de las flaquezas de PHP.

Laravel tiene una sintaxis bonita, semántica y creativa, que le permite destacar entre la gran cantidad de frameworks disponibles para el lenguaje. Hace que PHP sea un placer, sin sacrificar potencia y eficiencia. Es sencillo de entender, permite mucho la modularidad de código lo cuál es bueno en la reutilización de código.

Beneficios de Laravel

1. **Incluye un ORM**: A diferencia de CodeIgniter, Laravel incluye un ORM integrado. Por lo cual no debes instalar absolutamente nada.
2. **Bundles**: existen varios paquetes que extienden a Laravel y te dan funcionalidades increíbles..
3. **Programas de una forma elegante y eficiente**: No más código basura o espaguetti que no se entienden, aprenderás a programar ‘con clase’ y ordenar tu código de manera de que sea lo más re-utilizable posible.
4. **Controlas la BD desde el código**: Puedes tener un control de versiones de lo que haces con ella. A esto se llaman migrations, es una excelente herramienta, porque puedes manejar todo desde tu IDE, inclusive montar datos en tus tablas.
5. **Da soporte a PHP 5.3**.
6. **Rutas elegantes y seguras**: Una misma ruta puede responder de distinto modo a un método GET o POST.
7. **Cuenta con su propio motor de platillas HTML**.

8. **Se actualiza facilmente desde la línea de comandos**: El framework es actualizable utilizando composer update y listo, nada de descargar un ZIP y estar remplazando.
9. **Cuenta con una comunidad activa que da apoyo rápido al momento de que lo necesitas**.

Requerimientos iniciales

Para empezar a trabajar con Laravel es necesario cumplir con los siguientes requisitos iniciales:

- Un entorno de desarrollo web: Apache, IIS, Nginx PHP 5.3 o superior
- Base de datos: MySQL, Sqlite, Postgresql o sqlserver
- Librerías php : Mcrypt

Composer es una herramienta para administración de dependencias en PHP. Te permite declarar las librerías de las cuáles tu proyecto depende o necesita y éste las instala en el proyecto por ti.

Composer no es un administrador de paquetes. Sí, él trata con "paquetes" o "librerías", pero las gestiona en función de cada proyecto y no instala nada globalmente en tu equipo, por lo cual solo administra las dependencias del mismo.

Composer usa un archivo dentro de tu proyecto de Laravel para poder administrar las dependencias el cual se llama: **composer.json**. Este usa un formato JSON el cual se explicará más adelante, un ejemplo de él se muestra e esta imagen:

Ahora, composer no se limita a su uso unicamente con proyectos Laravel, sino que en Laravel el uso de composer nos facilita el control de dependencias y en la actualización de cada una como se explicó anteriormente. Para este curso se trabajará con este archivo pues es el que se va a crear al momento de instalar Laravel.

En este archivo podemos observar cierto orden en el acomodo de la información.

- **"name":** En esta sección se describe el nombre del usuario propietario del proyecto seguido del nombre del repositorio que aloja el proyecto separados por una barra(/).
- **"description":** Sirve para facilitar una breve descripción del paquete. Debemos ser muy claros y breves si deseamos colocar una descripción de nuestro paquete.
- **"keywords":** Estas palabras claves son una matriz de cadenas usadas para representar tu paquete. Son similares a etiquetas en una plataforma de blogs y, esencialmente, sirven al mismo propósito. Las etiquetas te ofrecen metadatos de búsqueda para cuando tu paquete sea listado en un repositorio.
- **"homepage":** La configuración de la página es útil para paquetes que van a ser de código libre. Puedes usar esta página para el proyecto o quizá para la URL del repositorio. Lo que creas que es más informativo.
- **"license":** Si tu paquete está pensado para ser redistribuido, querrás ofrecer una licencia con él. Sin una licencia muchos programadores no podrán usar el paquete por restricciones legales. Escoge una licencia que se ajuste a tus requisitos, pero que no sea muy restrictiva para aquellos que esperan usar tu código. El proyecto de Laravel usa la licencia MIT que ofrece gran libertad.
- **"authors":** ofrece información sobre los autores del paquete, y puede ser útil para aquellos usuarios que quieran contactar con el autor o autores. Ten en cuenta que la sección de autores permite una matriz de autores para paquetes colaborativos.

Gestor de dependencias

Una de las opciones interesantes del archivo composer.json es el campo **“require”**, en el se agregan como un arreglo el nombre de los paquetes que queremos incluir en nuestro proyecto seguido de la versión de cada dependencia.

Al final cuando se han agregado todas las dependencias que queremos para nuestro proyecto entonces solo basta con usar el siguiente comando en nuestra consola:

```
composer install
```

Con esto le indicamos a composer que debe descargar nuestras dependencias y las dependencias de estas dependencias para satisfacer las necesidades de nuestro proyecto. Para más información sobre composer, sus campos y su forma de uso podemos consultar su página oficial https://getcomposer.org/doc/ la cuál se encuentra en inglés.

Aprender más sobre HTML5

Para profundizar un poco más en HTML5 es recomendable el tutorial de w3schools.

Preparando nuestro entorno de trabajo.

Laravel necesita un servidor web. No importa cuál sea pero la mayoría de la comunidad usa Apache o Nginx y hacer lo mismo te pondrá las cosas más fáciles a la hora de buscar ayuda si la necesitas.

Instalación de XAMPP (Windows)

XAMPP es un programa que nos ofrece una distribución de Apache, MySQL, PHP y Perl muy simple de instalar, administrar y utilizar. Podemos descargarlo aquí.

Instalación de LAMP (Linux)

LAMP es el conjunto de aplicaciones Apache, MySQL, PHP o Python en entornos Linux que nos facilitan el desarrollo de sistemas.

En Ubuntu o derivadas podemos instalarlo con los siguientes comandos:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install lamp-server^
sudo apt-get install php5-mcrypt
sudo php5enmod mcrypt
```

Despues de tener instalado nuestro Servidor web, es necesario instalar composer el cuál es un gestor de dependencias php muy útil y del cuál se hablará más tarde.

Instalación de composer (Windows)

La forma más sencilla de instalar Composer en tu ordenador Windows consiste en descargar y ejecutar el archivo Composer-Setup.exe, que instala la versión más reciente de Composer y actualiza el PATH de tu ordenador para que puedas ejecutar Composer simplemente escribiendo el comando composer.

Instalación de composer (Linux)

En ubuntu bastará con ejecutar los siguientes comandos en la terminal.

```
sudo apt-get install curl
curl -sS https://getcomposer.org/installer | php
sudo mv composer.phar /usr/local/bin/composer
sudo echo 'PATH=$PATH:~/.composer/vendor/bin' >> ~/.profile
```

Instalación de Laravel

Existen diferentes formas de instalar laravel en nuestra computadora.

- Podemos clonar el repositorio Laravel de github.
- Usando el instalador:

  ```
  composer global require "laravel/installer=~1.1"
  laravel new Proyecto
  ```

- Usando composer:

  ```
  composer create-project laravel/laravel --prefer-dist Proyecto
  ```

  Una vez instalado laravel es recomendable situarse en la raíz del proyecto y ejecutar:

  ```
  composer update
  php artisan key:generate
  php artisan app:name Curso
  ```

PSR-4 y namespaces

¿Qué es PSR-4?

Es una especificación para la auto carga de clases desde la ruta de los archivos. Describe dónde se encuentran ubicados los archivos que serán autocargados. PSR-4 hace uso de namespaces para distinguir una clase de otra, esto es de gran ayuda cuando ocupamos librerías de terceros porque en muchas ocaciones existirán clases con el mismo nombre que las nuestras y podrían sobreescribirse o usar una que no queremos.

PSR-4 fue creada por el grupo de interoperabilidad de PHP, ellos han trabajado en la creación de especificaciones de desarrollo para este lenguaje para que estandarizemos diferentes procesos, como es en este caso el como nombrar las clases de nuestro proyecto y hacer uso de ellas.

Usar especificaciones PSR-4 no es obligatorio y su uso puede ser completo o parcial, aunque es recomendable no omitirlo porque a Composer le permite cargar nuestras clases automaticamente.

¿Qué es un autoloader?

Aparecieron desde la versión de PHP5 y nos permite encontrar clases para PHP cuando llamamos las funciones new() o class_exists(). De esta forma no tenemos que seguir haciendo uso de require() o include().

PSR-4 nos permite definir namespaces de acuerdo a la ruta de los archivos de las clases, es decir, si tenemos una clase "Pdf" en el directorio Clases/Templates/ , ese será su namespace. Podemos hacer un simil con el import de java.

El namespace de Clases/Templates quedaría de la siguiente forma: Clases\Templates\Pdf.php

Para usar PSR-4 en composer podemos definir el namespace de nuestra aplicación y el directorio dónde serán alojadas las clases, por ejemplo:

```
{
    "autoload":{
        "psr-4":{
            "Taller\\": "app/"
        }
    }
}
```

• • •

Hasta aquí la previsualización.