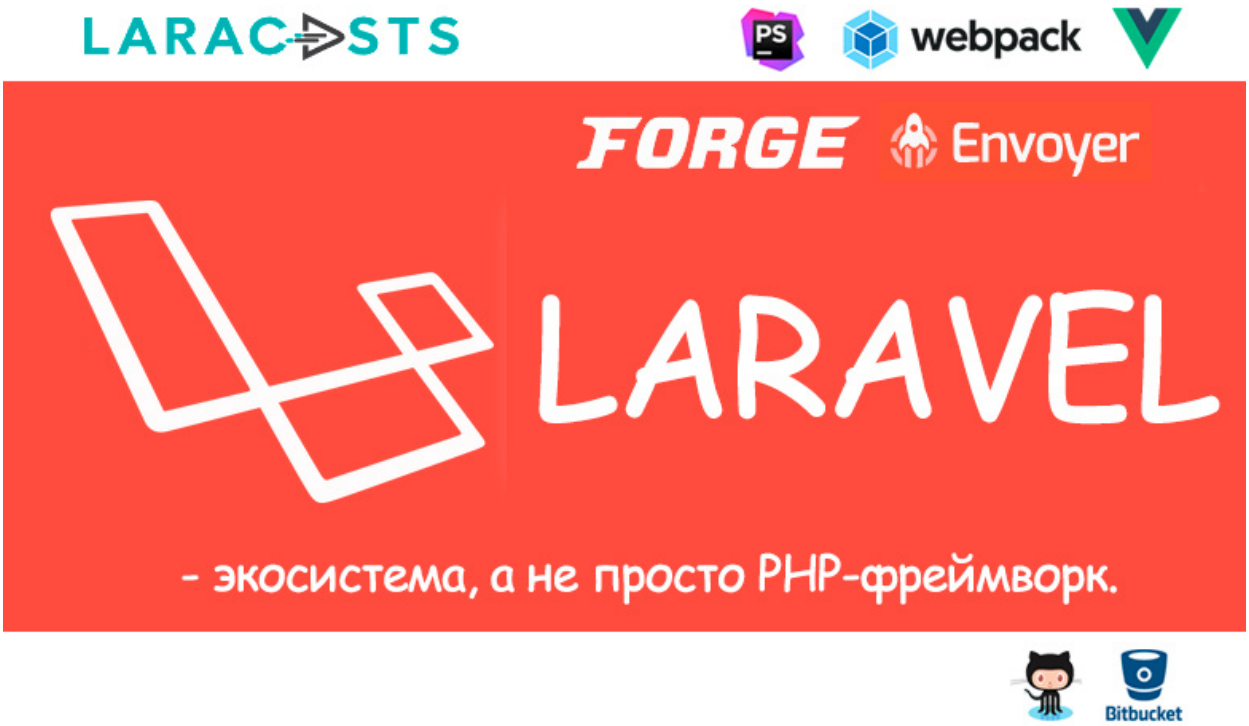
Laravel — экосистема, а не просто PHP-фреймворк

habrahabr.ru/post/334776/

SwVZFSf4fLt8NSZNfV87 3 августа 2017 в 12:27

- Из песочницы



Данная статья предназначена для начинающих веб-разработчиков, а также тех, кто хочет понять, для чего стоит изучить PHP-фреймворк Laravel и какую экосистему он нам предлагает. Статья написана на момент актуальности ***Laravel*** *версии* ***5.4****,* ***в августе 2017*** *выйдет релиз* ***Laravel 5.5****, который предоставляет ещё больше возможностей.*

Содержание:

Введение в веб-разработку: что было раньше и что сегодня

В данной статье хотелось бы рассказать о том, что такое PHP-фреймворк Laravel и как благодаря этому фреймворку начинающий веб-разработчик может расширить свой кругозор, а также поднять свой культурный уровень разработки. Попытаемся раскрыть тему создания веб-проекта и помочь начинающим.

Инженер, программист или веб-разработчик?

Если 30 лет назад инженер и программист был один и тот же человек, и можно было с уверенностью сказать, что он даже мог починить электронику, то сегодня, благодаря техническому прогрессу и развитию технологий мы имеем множество языков программирования, специализаций и направлений, для изучения которых

одному человеку просто физически не хватит времени. Скорее всего Ваш сосед программист не в состоянии починить смартфон, хотя он может написать приложение для этого смартфона.

Сегодня существует очень много разработчиков с разным уровнем знаний: есть профессионалы с большим опытом работы и знанием как минимум нескольких языков программирования и культуры разработки, а есть новички с большими амбициями и желанием развиваться.

Не секрет, что PHP считается языком программирования для разработки, на котором необходим минимальный набор знаний. Это язык программирования с очень низким порогом вхождения.

Буквально любой может взять и тут же вывести строку на экран. Именно поэтому опытные разработчики на любых языках программирования считают PHP-разработчиков «ненастоящими» разработчиками, а PHP – «ненастоящим» языком программирования.

Но возможно ли создать на PHP серьёзный продукт и как доказать другим, что PHP можно доверять? Если Вы из тех людей, которые считают PHP «несерьёзным» языком программирования, то советую дочитать до конца и, скорее всего, Вы измените своё мнение.

Разработка любого программного продукта всегда требует нечто большего, чем просто знания самого языка разработки. Для его создания и поддержки разработчику необходимо множество других знаний и навыков.

Мы будем говорить о разработке веб-проекта и о том, что сегодня необходимо знать веб-разработчику для успешного запуска веб-проекта, а главное – я попытаюсь показать, что

Laravel – это идеальное решение для тех, кто хочет быстро и грамотно создать безопасный и надёжный веб-проект, при этом всегда оставаясь на пике технологий веб-разработки.

Начало создания веб-проекта

Прежде чем начать создавать веб-проект, необходимо задуматься не только над структурой проекта и его фишками, но и над процессом разработки и, главное, над своим рабочим окружением.
Сам Laravel хоть и является PHP–фреймворком, но не стоит его недооценивать, ведь это целая экосистема для веб-разработки.

Если Вы скачали фреймворк Laravel в **.zip** архиве, чтобы оценить его возможности, тогда Ваше рабочее окружение точно настроено неверно, и у Вас существует пробел в знаниях. Вам однозначно стоит внимательно прочитать данную статью до конца.

Этап первый – процесс написания кода

Вы можете работать на любой операционной системе, в том числе и на Windows. Нам необходима хорошая **IDE** (Интегрированная среда разработки (англ. Integrated Development Environment)) – рекомендую **PhpStorm**. Можете использовать текстовый редактор **Atom** или **Sublime Text**. Конечно, можно писать код и в обычном блокноте, например, **Notepad++**, но хорошая IDE – незаменимая вещь.
Также после установки IDE или текстового редактора я советую потратить время и изучить как можно больше его возможностей и функций. Это поможет сэкономить массу времени в дальнейшем, а также автоматизировать многие рутинные задачи.

Многие считают, что «крутые» разработчики должны писать код в блокноте, но помнить по памяти названия функций – это одно, а не делать опечатки в коде, упростить и ускорить процесс разработки – это совсем другое. Главная задача – освоить все возможности IDE.

Кроме IDE нам необходимо будет установить **Composer**, именно через него мы и будем устанавливать (обновлять) Laravel, добавлять (обновлять) дополнительные пакеты в наш веб-проект.

Обязательно изучите работу с Composer, это очень важный и полезный инструмент.

Подробно изучите инструкцию по установке Laravel по этой ссылке.

Далее мы не будем описывать процесс написания кода, а предположим, что Вы уже установили IDE и Laravel.

После установки Laravel в коде сразу прописано отображение базовой страницы – этого достаточно, чтобы перейти к следующей части статьи.

Этап второй – тестирование кода

Для тестирования веб-проекта Вам не надо загружать файлы на **FTP**-сервер, устанавливать локальный **Apache** (тот же **Denwer** или **XAMPP**) – так делали много лет назад, а многие новички так делают до сих пор. Это неправильно и не спасёт от ошибок в коде. На сегодняшний день для этих задач есть соответствующие инструменты, которые сэкономят много времени и нервов.
Laravel предлагает нам установить Homestead.

Homestead – это образ операционной системы Ubuntu, в которой уже установлено всё необходимое.

С процессом установки и настройки Homestead Вы можете ознакомиться по ссылке.

Для установки образа нам понадобится **Vagrant** и **VirtualBox**. Благодаря данному образу Вы точно будете знать, какие модули надо установить и как поведёт себя Ваш код на Ubuntu. Вы также можете установить любой дополнительный софт.

Если кратко, то у Вас в системе появятся общие папки с кодом, которые будут доступны внутри образа Ubuntu, и выполняться Ваш код будет именно внутри Ubuntu.

В браузере Вы набираете **site.app**, и у Вас отображается сайт из Ubuntu. При этом у Вас также будет доступ к Ubuntu по SSH.

У начинающих установка и настройка Homestead займёт время, но как разработчик Вы просто обязаны это сделать.

Стоит отметить, что Homestead можно установить не только на Linux, но и на Windows.

Далее будем считать, что Homestead установлен, и сайт со свежей версией Laravel открывается у Вас в браузере.

Ваш код запускается в браузере, но действительно ли всё работает?

Ни один уважающий себя разработчик не должен писать код без тестов. Тесты позволяют нам быть точно уверенными, что всё работает так, как мы задумали. Не жалейте времени на написание тестов. Каждый профессиональный разработчик обязательно пишет тесты своего кода.

Laravel предлагает нам инструменты для полноценного тестирования веб-проекта со всех сторон. Вы можете тестировать всё: создать временную базу данных, проверить заполнение HTML-форм, проверить загрузку файлов, даже содержание PHP-сессий и отправку писем.

Laravel создан для качественного тестирования всех возможностей Вашего проекта.

Документацию по тестированию можно найти по этой ссылке.

В Laravel тесты находятся в папке **tests** и выполняются командой **phpunit** в консоле, либо сразу из IDE.

Тесты бывают нескольких типов:

1. Функциональные – Feature-тесты
2. Модульные – Unit-тесты

Feature-тесты – функциональные тесты.
Тесты, которые проверяют функционал веб-проекта, например: регистрацию пользователей, отправку уведомлений, заполнение веб-форм, загрузку файлов. Они позволяют нам проверить, какие именно данные отображаются в браузере. Теперь Вам не надо заполнять веб-формы вручную, чтобы узнать работают ли они.

Также Вы можете проводить тестирование с помощью Laravel Dusk, не просто отправляя HTTP-запросы, а используя реальный движок браузера Chromium.

В этом нам поможет **Laravel Dusk**.

Unit-тесты – модульные тесты.

Другой тип тестирования называется unit-тестированием. Эти типы тестов проверяют логику нашего приложения, каждую функцию, отлавливают события, определяют отправлено ли письмо, а также сверяют текст письма, проверяют добавлено ли задание в очередь сообщений и всё, что может сломаться, если Вы или кто-то ещё неудачно измените Ваш код.

Каждая функция проекта должна иметь свои тесты, а когда Вы завершите проект, то все тесты должны успешно запускаться.

При изменении функционала Вы можете дописать тесты. Это спасёт Вас и Ваших коллег от ошибок и поможет проще диагностировать проблему.

Unit-тестирование позволяет избежать ошибок в логике приложения.

Стоит отметить, что существует методика разработки **TDD** (test-driven development) – разработка через тестирование. Сначала мы пишем тесты, а затем постепенно реализуем код. Когда все тесты выполнены, то мы можем сказать, что завершили написание кода.

Если Вы ещё не писали тесты для своих проектов, значит пора переходить на новый уровень. Кроме тестов есть ещё другие помощники для анализа производительности веб-приложения.

Laravel предлагает нам установить **Laravel Debugbar**.

Это специальный пакет, который отображается на Вашем сайте в режиме отладки. С помощью него можно отследить все SQL-запросы к Вашей базе данных с целью их дальнейшей оптимизации.

Этап третий – сборка проекта

После создания веб-проекта и прохождения тестов нам необходимо подготовить наш проект к размещению на сервере.
Laravel предоставляет нам **Laravel Mix**.

Laravel Mix использует **Webpack** и умеет работать с **CSS, JS, Less, Saas, Stylus, PostCSS**.

Это замечательный инструмент, который, используя специальный сборщик модулей Webpack, собирает вместе все наши JS и CSS-файлы, а также, самое главное, умеет создавать версии этих файлов.

Таким образом, каждая сборка нашего проекта позволяет иметь разные названия JS и CSS-файлов в HTML-коде, что решает проблему с кешированием при изменении содержимого файла.

В шаблоне нашего проекта пишем:

```
<link rel="stylesheet" href="{{ mix('/css/app.css') }}">
```

После сборки он превращается в:

```
<link href="/css/app.289df32d2d2c47df3b16.css" rel="stylesheet">
```

При этом браузер посетителя сразу загрузит новый файл с сайта.
Не правда ли, удобно? Точно также и с JS-файлами.

Стоит отметить, что Laravel замечательно работает с прогрессивным Javascript-фреймворком Vue и позволяет очень удобно создавать веб-приложения на базе этого JS-фреймворка. При этом каждый компонент можно удобно размещать в отдельном файле.

О том, как писать компоненты для Vue используя Laravel можно прочитать по этой ссылке.

Этап четвёртый – развёртывание (deploy) кода

Обычно после сборки проекта его файлы необходимо загрузить на сервер и обновить структуру таблиц в базе данных.
~~Берём папку с файлами и загружаем на FTP-сервер.~~
~~Заходим в phpMyAdmin и делаем изменения в БД.~~

Мы не станем использовать **FTP** и **phpMyAdmin**, иначе пока мы вносим изменения, все пользователи, которые зайдут на сайт веб-проекта, увидят множество ошибок об отсутствии каких-то файлов или полей в БД.

Мы можем, конечно, объявить о проведении технических работ, показав нашим пользователям насколько технически сложен наш проект, что требует полного отключения, но никого это точно не обрадует.

Есть очень простое и грамотное решение, которое позволяет обновлять код веб-проекта без его отключения, и ни один пользователь при этом не получит сообщения об ошибке.
Первое что нам необходимо изучить — **Git**.

Git — это распределённая система управления версиями файлов.

С помощью Git можно отслеживать изменения в файлах, возвращать старую версию файлов и работать в команде над одним и тем же кодом, при этом ничего не перепутав.

Использовать Git можно через сервис.

Вы можете создать либо общедоступный код, либо приватный (для приватных репозиториев – он платный).

Также Вы можете использовать другой бесплатный сервис BitBucket, который позволяет бесплатно создавать приватные репозитории с кодом.

Кроме этого, сам Git можно настроить так, чтобы при внесении изменений происходили определённые действия:

Таким образом, весь код Вашего веб-проекта будет храниться в Git, он всегда будет **качественный** и **проверенный**.
Например, если Вы предложите внести изменения в официальный код PHP-фреймфорка Laravel, то при внесении изменений автоматически запускаются тесты, которые проверяют работу фреймворка, учитывая новый код.

Ранее мы говорили о процессе развёртывания веб-приложения. Именно для этого нам и необходим Git. С Вашей локальной машины Вы загружаете код веб-приложения в Git, после чего произойдёт автоматический запуск развёртывания приложения на сервере.

Laravel Forge – сервер без хлопот. Для автоматического развёртывания из Git нам поможет сервис Laravel Forge.

Через Laravel Forge Вы можете создать виртуальный сервер в **DigitalOcean**, **Linode** или указать доступ к своему собственному серверу. При этом будет настроено абсолютно всё необходимое ПО для работы PHP-фреймворка Laravel.

Laravel Forge автоматически устанавливает обновления, связанные с безопасностью системы. Также Forge легко установит бесплатный **SSL-сертификат** от **Let's Encrypt**.

Вы можете дать сервису Laravel Forge доступ к Вашему Git-репозиторию и при каждом изменении в коде на сервере будет автоматически развёрнута его свежая версия.
Хотите 10 серверов? – Без проблем, Laravel Forge может установить **балансировщик** нагрузки, создать 10 виртуальных серверов, на каждый сервер копировать код из Git и запустить проект.

Думаете всё?

Нет, совместно с **Envoyer** Вы можете запускать новый код в работу без остановки сервиса совсем.

Хотя лично я не использую Envoyer, а просто написал небольшой скрипт в панели Laravel Forge, который запускается при каждом развёртывании кода и обеспечивает замену на лету, при этом сохраняя ещё несколько копий старого кода на самом сервере.

→ Ссылка на скрипт

Итоги

Мы создали комфортное рабочее окружение, установили IDE, Composer, PHP-фреймворк Laravel, написали код проекта, запустили тесты, изучили систему

контроля версий Git, отправили туда код, подключили сервис Laravel Forge, при желании подключили также Laravel Envoyer, сделали развёртывание проекта на рабочий сервер из нашего Git-репозитория.
Можно сказать, что Laravel направил нас на грамотный путь в веб-разработке. Впереди ещё многое предстоит изучить, но мы уже проделали большую работу и можем начинать работать в команде с другими разработчиками.

Основные возможности PHP-фреймворка Laravel

А теперь рассмотрим возможности самого PHP-фреймворка Laravel: какие веб-приложения позволяет нам создавать данный PHP-фреймворк, насколько он продвинутый в техническом плане и почему он так популярен во всём мире.
После выхода **PHP7** по сравнению с **PHP5**, скрипты стали быстрее и начали использовать гораздо меньше оперативной памяти, а в связке с **Zend OPCache** показывают замечательные результаты. В частности сервис Laravel Forge настраивает Zend OPCache для достижения максимальной производительности.

Именно поэтому, когда идёт речь о производительности того или иного PHP-фреймворка, то всегда проводят тестирование без кеширования, работы с БД или файлами, в основном совершая множество вызовов к обычной PHP странице. В этом плане данный PHP-фреймворк существенно ничем не отличается от всех остальных, но когда речь идёт о масштабируемости, гибкости, универсальности встроенных механизмов кеширования и скорости разработки, именно тогда Laravel показывает всю свою гибкость и мощь.

Сам Laravel постоянно совершенствуется и следует современным тенденциям. Изучая его, Вы не отстанете от мира веб-разработки, главное – не зацикливаться на какой-то конкретной версии фреймворка, а развиваться вместе с ним. Для этого необходимо также изучать нововведения Laravel.

Ежегодно проводятся различные конференции, которые можно посмотреть также и онлайн.

Постараюсь описать основные возможности Laravel, чтобы можно было оценить масштаб:

- **MVC** (англ. Model View Controller – модель-представление-контроллер) PHP-фреймворк построен на базе известных и надёжных компонентов **Symfony**.
- Необходимые модули для фреймворка подключаются в виде пакетов-провайдеров (service provider). В версии Laravel 5.5 достаточно просто установить пакет через Composer, и он сразу будет доступен, без необходимости что-либо писать в коде.
- Код фреймворка отделён от кода разработчика, каждый компонент легко расширяется.
- Код веб-проекта, CSS, JS, HTML-код страниц разделены в отдельные директории. Фреймворк использует замечательный шаблонизатор Blade, который позволяет отделить вёрстку от PHP-кода. Сам шаблонизатор настолько прост, что даже начинающий HTML-верстальщик сможет легко его

осилить.
- Удобная маршрутизация, валидация входящих параметров.
- Кеширование, работа с хранилищами файлов, работа с различными БД.
- Миграции для базы данных, Вы можете изменять структуру БД и откатывать изменения.
- Очереди заданий, планировщик задач, консоль, работа с SSH.
- Огромный функционал **Eloquent ORM** позволяет полностью обезопасить себя от атак типа **SQL Injection**, а также загружать данные из нескольких таблиц (решая проблему **N+1**) или же обрабатывать данные из БД частями.
- **Laravel Collections** – можно сказать, что это PHP массивы, но с очень продвинутыми возможностями, которые экономят массу времени.
- Кэширование файлов маршрутизации, файлов конфигурации, шаблонов. Это ускоряет работу фреймворка.
- Отправка уведомлений различными способами: почта, **Slack** и т.д., можете дописать сами.
- Поддержка **WebSockets** для создания настоящих интерактивных приложений.
- Поддержка мультиязычности: легко добавляйте любые языки, а пакет **Laravel-lang** уже содержит множество переводов.
- Интерфейс командной строки **artisan**, который позволяет генерировать модели, контроллеры, уведомления, запускать задания из очереди заданий и многое другое.
- **Laravel Tinker** – дополнительный пакет, который позволяет работать с кодом проекта из командной строки.
- Огромные возможности для тестирования веб-проекта, включая заполнение базы данных тестовыми данными.
- У фреймворка есть даже собственный сайт с библиотекой пакетов.
- Нужен полнотекстовый поиск? Пожалуйста – **Laravel Scout**, можно использовать **Algolia**, **Sphinx** и другие драйвера.

Впечатляет, не правда ли? А я не описал даже и половины возможностей.
С помощью Laravel можно одной командой сгенерировать систему регистрации и входа на сайт и с лёгкостью подключить сервисы **OAuth** аутентификации благодаря **Laravel Socialite** или даже создать свой с помощью **Laravel Passport**.

Для тех, кто не знает OAuth, – это возможность войти на сайт через социальные сети.

Это лишь малая часть того, что умеет Laravel, и если начинающий PHP-разработчик изучит его возможности, даже не углубляясь в ядро самого PHP-фреймворка, то это безусловно поднимет его уровень знаний не только в разработке на Laravel, а в веб-разработке в частности, включая навыки работы в команде и понимание принципов разработки высоконагруженных проектов.

На основном сайте PHP-фреймворка Laravel недаром присутствует девиз:

> «Любите красивый код? Мы тоже. PHP-фреймворк для веб-мастеров.»

Ведь код PHP-фреймворка Laravel не только красивый, приятно читаемый, но ещё и очень грамотно продуман, а над любым изменением думает множество людей, что позволяет создавать профессиональные веб-приложения на уровне мастера своего дела.

Полезные ссылки:

Сайт PHP-фреймворка Laravel.
Основные новости PHP-фреймворка и новости о различных пакетах.

Laravel на русском и русскоязычное сообщество Laravel в VK.

Очень рекомендую сайт https://laracasts.com, где **Jeffrey Way** в своих видео-уроках наглядно и без лишних слов показывает возможности Laravel, также рассказывает много полезных вещей. За 2 минуты человек успевает рассказать больше и доступнее, чем многие за час.

А также рекомендую книгу ***"Refactoring to Collections"***, где **Adam Wathan** подробно рассказывает о возможностях Laravel Collections. Гарантирую, Ваш код изменится в лучшую сторону.

Рекомендую в каждый веб-проект на Laravel устанавливать:

P.S.: Данная статья получилась довольно объёмной и без технических подробностей, но её основная задача – дать начинающим разработчикам вводные знания, объяснить что же такое Laravel и какие возможности он даёт, а также показать, что Laravel – это не просто PHP-фреймворк, а целая экосистема, которая постоянно развивается. Это именно тот тренд, на который должны обратить внимание PHP-разработчики.
Многие вещи могут быть непонятны для начинающих, но не стоит отчаиваться. Любой термин достаточно быстро можно найти в сети. Я лишь попытался обобщить информацию именно в том порядке, в котором должен её воспринимать начинающий разработчик.

Если Вам понравилась статья, то буду уже подробнее писать про каждый этап в данной статье с техническими деталями и кодом.

Метки: