ГЛАВА 1

Зачем использовать Laravel

На заре появления динамических веб-страниц процесс написания веб-приложения выглядел совершенно не так, как сейчас. Разработчик должен был не только реализовать уникальную бизнес-логику приложения, но и написать код всех других компонентов, которые так часто присутствуют на сайте, — модулей аутентификации пользователей, валидации ввода, доступа к базе данных, обработки шаблонов и т. д.

Сегодня у программистов есть десятки фреймворков для разработки приложений и тысячи легкодоступных компонентов и библиотек. Разработчики шутят, что к концу изучения одного фреймворка появляется три новых, которые предположительно лучше и должны его заменить.

В альпинизме наличие горы может быть достаточным основанием для ее покорения, однако, когда мы решаем, какой из фреймворков лучше использовать и стоит ли вообще это делать, нужно исходить из более веских оснований. Необходимо спросить у себя, зачем применять фреймворк и, в частности, Laravel.

Для чего нужен фреймворк

Почему PHP-разработчикам выгодно использовать доступные отдельные компоненты или пакеты, вполне очевидно. При этом кто-то другой разрабатывает и поддерживает изолированный фрагмент кода. Этот человек с четко очерченной сферой ответственности и теоретически лучше вас разбирается в своем компоненте, поскольку вы можете уделить каждой части лишь очень ограниченное время.

Такие фреймворки, как Laravel, Symfony, Lumen или Slim, предварительно упаковывают коллекцию сторонних компонентов вместе со специфическим «клеем» фреймворка — файлами конфигурации, сервис-провайдерами, предписываемой структурой каталогов и программами начальной загрузки. Таким образом, преимущество от использования фреймворка в том, что кто-то другой принимает решение в отношении отдельных компонентов и того, *как они должны сочетаться*.

Все своими руками

Допустим, вы решили создать веб-приложение без использования фреймворка. С чего начать? Вероятно, приложение должно осуществлять маршрутизацию HTTP-запросов, и потому нужно оценить доступные библиотеки HTTP-запросов и ответов и выбрать одну из них. Вам также следует выбрать маршрутизатор. И конечно, вам необходимо настроить некий файл конфигурации маршрутов. Какой синтаксис следует использовать? Где его разместить? А что насчет контроллеров? Где их разместить и как должна производиться загрузка? Надо использовать контейнер внедрения зависимостей для разрешения доступа к контроллерам и их зависимостей. Но какой?

Более того, ответив на все эти вопросы и успешно создав приложение, подумайте, как ваши решения отразятся на следующем разработчике. Представьте, что вам нужно поддерживать четыре или десять таких приложений на базе собственного фреймворка, помнить расположение контроллеров и синтаксис маршрутизации в каждом из этих приложений!

Согласованность и гибкость

Фреймворки решают эту проблему продуманным ответом на вопрос о том, какой компонент следует использовать для разных целей, и гарантией хорошей совместимости выбранного набора компонентов. Благодаря тому что фреймворки подразумевают выполнение ряда соглашений, разработчик разбирается в меньшем количестве кода при переходе к новому проекту, если вы знаете, как осуществляется маршрутизация в одном проекте Laravel, то вы знаете, как она работает во всех.

Когда кто-то советует вам создавать фреймворк для каждого проекта, подразумевается, что вы должны *контролировать*, что следует включать, а что — нет в «фундамент» вашего приложения. Хороший фреймворк не только обеспечит вам надежный «фундамент», но и позволит настраивать его состав. Как вы увидите далее, популярность Laravel во многом обусловлена именно такой возможностью.

Краткий экскурс в историю веб- и PHP-фреймворков

Чтобы ответить на вопрос, зачем использовать Laravel, нужно вспомнить, как появился этот фреймворк и что происходило до этого. Популярности Laravel предшествовало появление множества разнообразных фреймворков и ряд иных тенденций в сфере разработки на PHP и в других областях веб-разработки.

Ruby on Rails

В 2004 году Дэвид Хайнемайер Хансон выпустил первую версию Ruby on Rails, и практически каждый фреймворк для веб-приложений, выпущенный после, имел какие-то черты Rails.

Фреймворк Rails популяризировал MVC, API на базе RESTful и JSON, программирование по соглашениям, шаблон Active-Record и многие другие инструменты и соглашения, которые значительно повлияли на подход веб-программистов к созданию приложений, в частности увеличили скорость разработки.

Бум PHP-фреймворков

Большинство разработчиков понимало, что будущее за такими веб-фреймворками, как Rails, и на свет как грибы после дождя стали появляться новые PHP-фреймворки, в том числе откровенные копии Rails.

Первой ласточкой стал появившийся в 2005 году CakePHP, за которым последовали Symfony, CodeIgniter, Zend Framework и Kohana (ответвление от проекта CodeIgniter). В 2008 году вышел фреймворк Yii, а в 2010 году — Aura и Slim. В 2011 году появились FuelPHP и Laravel, которые не были простыми ответвлениями CodeIgniter, а предлагались в качестве альтернативы.

Многие из них сильно напоминали Rails, будучи ориентированными на применение объектно-реляционных отображений (ORM), MVC-структур и иных инструментов быстрой разработки. Другие, как, например, Symfony и Zend, были больше направлены на использование корпоративных шаблонов проектирования и электронной коммерции.

Преимущества и недостатки CodeIgniter

Фреймворки CakePHP и CodeIgniter были первыми в ряду PHP-фреймворков, созданных под влиянием Rails. CodeIgniter быстро завоевал известность, и к 2010 году стал, пожалуй, самым популярным независимым PHP-фреймворком.

Фреймворк CodeIgniter был прост, удобен в использовании, имел отличную документацию и мощное сообщество. Однако он медленно развивался в плане использования современных технологий и шаблонов и по мере прогресса фреймворков и совершенствования инструментов PHP-разработки начал отставать и с точки зрения технологий, и с точки зрения предлагаемых по умолчанию функциональных возможностей. В отличие от многих других фреймворков, CodeIgniter разрабатывался компанией, поэтому медленно внедрялась поддержка новых возможностей PHP 5.3, таких как пространства имен или использование GitHub, а позднее — Composer. К 2010 году создатель Laravel Тейлор Отвел настолько разочаровался в CodeIgniter, что решил написать собственный фреймворк.

Laravel 1, 2 и 3

Релиз бета-версии Laravel 1, написанной с нуля, состоялся в июне 2011 года. В ней присутствовали собственная ORM-система (Eloquent), маршрутизация на основе замыканий (навеянная фреймворком Ruby Sinatra), возможность расширения за счет использования модульной системы и хелперы (helper, вспомогательные функции) для форм, валидации, аутентификации и т. д.

Поначалу разработка Laravel велась очень быстро, Laravel 2 и 3 появились соответственно в ноябре 2011-го и феврале 2012 года. В них добавили контроллеры, модульное тестирование, инструмент командной строки, контейнер инверсии управления (Inversion of Control, IoC), отношения Eloquent и миграции.

Laravel 4

При создании Laravel 4 Тейлор полностью переписал весь фреймворк. К этому моменту Composer, менеджер пакетов для PHP, уже распространился настолько, что стал практически промышленным стандартом, и Отвел решил, что будет полезно переписать фреймворк в виде набора компонентов, которые можно распространять и упаковывать с помощью этого пакетного менеджера.

Тейлор разработал его под кодовым названием *Illuminate* и в мае 2013 года выпустил Laravel 4 с совершенно новой структурой. Теперь вам уже не предлагался для загрузки пакет с большей частью кода фреймворка; вместо этого Laravel подгружал большинство своих компонентов из фреймворка Symfony (компоненты которого можно свободно использовать в других фреймворках) и набора компонентов Illuminate с помощью менеджера Composer.

В Laravel 4 были также введены очереди, почтовый компонент, фасады и механизм заполнения баз данных первичными данными. Поскольку в Laravel теперь использовались компоненты фреймворка Symfony, разработчики объявили, что релизы Laravel будут «зеркалировать» (с небольшим отставанием) шестимесячный график выпуска фреймворка Symfony.

Laravel 5

В ноябре 2014 года планировалось выпустить Laravel 4.3, но по мере продвижения разработки стало ясно, что изменения стоит выделить в более крупный релиз, которым стала версия Laravel 5 в феврале 2015 года.

В Laravel 5 была обновлена структура каталогов, удалены вспомогательные функции для форм и HTML, введены контрактные интерфейсы, множество новых представлений, библиотека Socialite для аутентификации в социальных сетях, библиотека Elixir для компиляции ресурсов, библиотека Scheduler для упрощения планирования задач cron, библиотека dotenv для более легкого управления средой, запросы форм и совершенно новая реализация REPL-интерфейса (read —

evaluate — print loop, цикл «чтение — вычисление — вывод»). С тех пор фреймворк рос в плане функциональности и зрелости, но никаких серьезных изменений, как в предыдущих версиях, уже не вносилось.

Laravel 6

В сентябре 2019 года вышла версия Laravel 6, содержащая два основных изменения: во-первых, были удалены глобальные вспомогательные функции для работы со строками и массивами (и отдано предпочтение фасадам) и, во-вторых, был завершен переход на SemVer (семантическое управление версиями) для нумерации версий. Благодаря этому новые версии после Laravel 5 — как основные (6, 7 и т. д.), так и второстепенные (6.1, 6.2 и т. д.) — стали выпускаться гораздо чаще.

Версии Laravel в новом мире SemVer (6+)

Начиная с версии 6, выпуски Laravel стали менее монументальными, чем раньше, благодаря новому графику выпуска SemVer. И в дальнейшем выпуск новых версий будет зависеть от того, сколько времени прошло с момента выхода предыдущей версии, а не от факта реализации заметных новых возможностей.

Чем уникален Laravel

Так что же делает Laravel уникальным? Почему для работы лучше выбрать его, а не какой-либо другой PHP-фреймворк? Ведь в каждом из них используются компоненты Symfony? Немного поговорим о том, что делает Laravel «именно тем, что нужно».

Философия Laravel

Чтобы понять, что выгодно отличает данный фреймворк от других, достаточно прочитать его рекламные материалы и файлы README. Обратите внимание, какие слова употребляет Тейлор: сначала связанные со светом — «освещать», «искра», затем — «искусные мастера», «элегантные», а также «дыхание свежего воздуха», «новый старт» и, наконец, — «быстрый», «сверхсветовая скорость».

Двумя принципами данного фреймворка являются повышение скорости разработки и удобства разработчиков. Тейлор специально назвал интерфейс командной строки Artisan («искусный мастер»), чтобы подчеркнуть контраст с более утилитарными ценностями. Зачатки этого мышления прослеживались еще в 2011 году, когда в одном из своих вопросов в сети StackExchange (https://oreil.ly/q0tgM) Отвел признался, что порой тратит невероятно много времени — целые часы — на то, чтобы придать коду «красивый» вид — лишь для того, чтобы ему было приятно на него смотреть. Кроме того, он часто говорил, как важно предоставить разработчикам

возможность проще и быстрее воплощать идеи, исключив ненужные препятствия для создания великолепных продуктов.

По сути, Laravel призван снабдить разработчиков инструментами и возможностями. Его задача — предоставить понятный, простой и красивый код и функции, позволяющие программистам быстро изучать, запускать, разрабатывать и писать код, отличающийся простотой, ясностью и долговечностью.

Принцип ориентации на разработчиков ясно прослеживается во всех материалах по Laravel. «Счастливые разработчики пишут лучший код», — написано в документации. Одно время неофициальный рекламный слоган фреймворка звучал как «Счастье разработчика от загрузки до развертывания». Конечно, создатели любого инструмента или фреймворка могут сказать, что они заботятся о счастье программистов. Однако только в Laravel этому отводится *центральное* место, что оказало огромное влияние на общий стиль и процесс принятия решений. Если в других фреймворках на первое место может ставиться архитектурная чистота или совместимость с целями и ценностями корпоративных групп разработчиков, в Laravel основное внимание уделяется нуждам и запросам отдельного разработчика. Это совсем не значит, что вы не сможете писать архитектурно чистые или корпоративные приложения с помощью Laravel; это лишь значит, что вам не придется жертвовать читабельностью и понятностью кодовой базы.

Как Laravel делает разработчиков счастливее

Легко сказать, что вы хотите сделать разработчиков счастливыми, и гораздо труднее претворить это в жизнь. Для этого нужно ответить на вопрос, что во фреймворке может осчастливить программистов или огорчить. В Laravel применяется несколько подходов, призванных облегчить жизнь разработчиков.

Во-первых, Laravel — это фреймворк для быстрой разработки. Он предельно прост в освоении и сводит к минимуму количество шагов между запуском нового приложения и его публикацией. Его компоненты упрощают разработку веб-приложения: от взаимодействия с базой данных и аутентификации до работы с очередями, электронной почтой и кэшем. Компоненты Laravel не только великолепно справляются со своей задачей, но и предоставляют единообразные API и предсказуемые структуры в рамках всего фреймворка: когда вы пробуете что-то новое в Laravel, вам нужно просто взять и запустить новую функцию.

Причем этот подход не ограничивается рамками фреймворка. Laravel предоставляет целую экосистему инструментов для создания и запуска приложений. У вас есть Sail, Homestead и Valet для локальной разработки, Forge для управления серверами и Envoyer с Vapor для расширенного развертывания. Имеется набор дополнительных пакетов: Cashier для платежей и подписок, Echo для веб-сокетов, Scout для поиска, Sanctum и Passport для API-аутентификации, Dusk для тестирования клиентской части приложения (фронтенда), Socialite для аутентификации в социальных сетях, Horizon для отслеживания состояния очередей, Nova для создания

панелей администратора и Spark для развертывания собственного SaaS-сервиса. Laravel призван избавить разработчиков от однообразных операций, чтобы они могли сосредоточиться на более творческих задачах.

В Laravel исповедуется принцип «программирования по соглашениям»: если вы согласитесь применять предлагаемые значения по умолчанию, то выполните гораздо меньше работы, чем при использовании других фреймворков, которые требуют объявления всех настроек даже с рекомендованной конфигурацией. Создание проекта в Laravel занимает меньше времени, чем в большинстве других PHP-фреймворков.

Большое внимание уделяется простоте. Вы можете внедрять и имитировать зависимости, шаблоны Data Mapper и репозитории, разделение ответственности команд и запросов (CQRS) и любые другие более сложные архитектурные шаблоны. Однако если другие фреймворки, как правило, предлагают задействовать эти инструменты и структуры при создании каждого проекта, Laravel, его документация и сообщество обычно изначально предлагают простейший вариант реализации на основе таких средств, как глобальная функция, фасад и ActiveRecord. Это позволяет создавать для решения задач предельно простое приложение, не отказываясь от возможности его применения в сложной среде.

Интересное отличие Laravel от других PHP-фреймворков в том, что его создатель и сообщество больше тяготеют к идеям Ruby on Rails и функциональным языкам программирования, чем к языку Java. В мире PHP очень сильна тенденция к многословности и сложности, характерная для наиболее Java-подобных аспектов PHP. Laravel же продвигает выразительные, динамичные и простые методы кодирования и возможности языка.

Сообщество Laravel

Если вам еще не приходилось сталкиваться с сообществом Laravel, то приготовьтесь открыть для себя нечто необычное. Одна из отличительных черт фреймворка Laravel, в немалой степени способствовавшая его росту и успеху, — доброжелательное и готовое делиться знаниями сообщество программистов. Начиная с видеоуроков Laracasts (https://laracasts.com/) от Джеффри Уэя, Laravel News (https://laravel-news.com/) и Slack-, IRC- и Discord-каналов, заканчивая постами в «Твиттере», блогами, подкастами и конференциями Laracon — все говорит о том, что у Laravel есть обширное и энергичное сообщество, в котором присутствуют и те, кто был в числе энтузиастов с самого первого дня, и те, кто только знакомится с этим фреймворком. И это не случайность.

> С самых первых дней фреймворка Laravel у меня была идея о том, что любой человек хочет чувствовать себя частью чего-то значимого. Это желание принадлежать к некоторой группе единомышленников заложено в нас самой природой. Поэтому, привнеся в веб-фреймворк немного личностного подхода и достаточно активно взаимодействуя с сообществом, можно усилить в нем это чувство.
>
> *Тейлор Отвел, интервью о продукте и поддержке*

Уже в самом начале работы над Laravel Тейлор понял, что для успеха проект с открытым исходным кодом должен обладать двумя вещами: хорошей документацией и доброжелательным сообществом. И сегодня их можно смело назвать отличительными чертами Laravel.

Как работает Laravel

До сих пор мы говорили с вами исключительно об абстрактных вещах. А как насчет кода, спросите вы? Рассмотрим простое приложение (пример 1.1), чтобы вы могли увидеть, что такое работа с Laravel.

Пример 1.1. Программа Hello, World! в файле routes/web.php

```
<?php

Route::get('/', function () {
   return 'Hello, World!';
});
```

Самое простое, что можно сделать в приложении Laravel, — определить маршрут и возвращать результат каждый раз, когда кто-то по нему переходит. Если вы инициализируете на компьютере новое приложение Laravel, определите маршрут из примера 1.1 и настроите работу сайта из *публичного* каталога, то получите полностью функциональную программу Hello, World! (рис. 1.1).

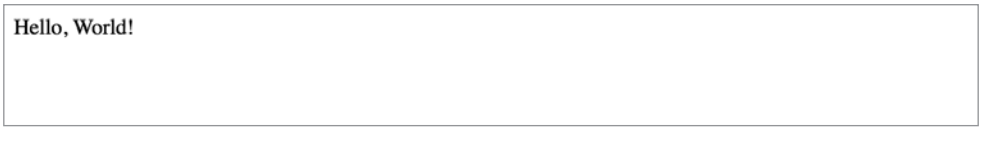

Рис. 1.1. Возвращение строки Hello, World! с помощью Laravel

Как показывает пример 1.2, реализация этой программы с помощью контроллеров выглядит очень похоже (если вы захотите сразу опробовать этот пример, то вам придется сначала создать контроллер командой `php artisan make:controller WelcomeController`).

Пример 1.2. Реализация программы Hello, World! с помощью контроллеров

```
// Файл: routes/web.php
<?php

use App\Http\Controllers\WelcomeController;

Route::get('/', [WelcomeController::class, 'index']);

// Файл: app/Http/Controllers/WelcomeController.php
<?php
```

```
namespace App\Http\Controllers;

class WelcomeController extends Controller
{
    public function index()
    {
        return 'Hello, World!';
    }
}
```

Эта программа будет выглядеть во многом так же и в том случае, если вы разместите несколько приветствий в базе данных (пример 1.3).

Пример 1.3. Программа Hello, World! с выводом нескольких приветствий из базы данных

```
// Файл: routes/web.php
<?php

use App\Greeting;

Route::get('create-greeting', function () {
    $greeting = new Greeting;
    $greeting->body = 'Hello, World!';
    $greeting->save();
});

Route::get('first-greeting', function () {
    return Greeting::first()->body;
});

// Файл: app/Models/Greeting.php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Greeting extends Model
{
    use HasFactory;
}

// Файл: database/migrations/2023_03_12_192110_create_greetings_table.php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
      * Запустить миграции
      */
    public function up(): void
```

```
    {
        Schema::create('greetings', function (Blueprint $table) {
            $table->id();
            $table->string('body');
            $table->timestamps();
        });
    }

    /**
     * Отменить изменения, произведенные в ходе миграции
     */
    public function down(): void
    {
        Schema::dropIfExists('greetings');
    }
};
```

Если пример 1.3 кажется вам немного сложным, пока пропустите его. Мы подробно разберемся с этим кодом в следующих главах. Как можно понять уже сейчас, чтобы настроить миграции базы данных и модели, а затем извлечь записи, требуется написать всего несколько строк кода — все предельно просто!

Почему стоит выбрать Laravel

Потому что Laravel позволит вам воплощать свои идеи в жизнь без напрасно написанного кода, в соответствии с современными стандартами кодирования, в среде активного сообщества и при наличии мощной экосистемы инструментов.

И потому что вы, дорогой разработчик, заслуживаете того, чтобы быть счастливым.