라라벨로 배우는
실전 PHP
웹 프로그래밍

라라벨로 배우는
실전 PHP 웹 프로그래밍

초판 1쇄 발행 2016년 11월 23일

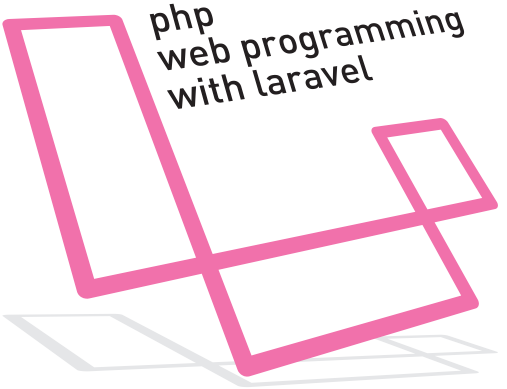
지은이 김주원
펴낸이 장성두
펴낸곳 제이펍

출판신고 2009년 11월 10일 제406-2009-000087호
주소 경기도 파주시 회동길 159 3층 3-B호
전화 070-8201-9010 / **팩스** 02-6280-0405
홈페이지 www.jpub.kr / **원고투고** jeipub@gmail.com
독자문의 readers.jpub@gmail.com / **교재문의** jeipubmarketer@gmail.com

편집부 이민숙, 황혜나, 이 슬, 이주원 / **소통·기획팀** 민지환, 현지환
표지디자인 미디어픽스
용지 에스에이치페이퍼 / **인쇄** 한승인쇄 / **제본** 광우제책사

ISBN 979-11-85890-62-3 (93000)
값 30,000원

라라벨로 배우는
실전 PHP 웹 프로그래밍



김주원 지음

Jpub 제이펍

※ 드리는 말씀

- 이 책에 기재된 내용을 기반으로 한 운용 결과에 대해 지은이, 소프트웨어 개발자 및 제공자, 제이펍 출판사는 일체의 책임을 지지 않으므로 양해 바랍니다.
- 이 책에 등장하는 각 회사명, 제품명은 일반적으로 각 회사의 등록 상표 또는 상표입니다. 본문 중에는 ™, ©, ® 마크 등이 표시되어 있지 않습니다.
- 이 책에서 사용하고 있는 제품 버전은 독자의 학습 시점이나 환경에 따라 책의 내용과 다를 수 있습니다.
- 책 내용과 관련된 문의사항은 지은이나 출판사로 연락해 주시기 바랍니다.

  – 지은이: juwonkim@me.com

  – 출판사: readers.jeipub@gmail.com

차례

APPENDIX 부록 _475

머리말

"졸면 죽는다"

2010년에 이 말을 처음으로 접했다. 모 이동 통신사 사장님이 입버릇처럼 외치고 다니셨던 경구를 이찬진 님이 트위터에서 자주 인용하셨다.

지난 15년 동안의 성공에 심취한 PHP 생태계는 잠깐 졸고 있었다. 웹 프로그래머들은 PHP의 미래가 없다고 생각하며 다른 언어로 전향하기 시작하는 등 PHP의 인기는 사그라지는 조짐을 보였다.

나를 비롯한 초기 웹 개발자들은 C나 펄(Perl)로 웹 개발을 했었다. HTML 태그 출력을 위해 printf 함수나 HEREDOC 문법을 이용해야 했고, 한 줄만 수정했더라도 매번 컴파일해서 결과를 확인해야만 했다. 이런 와중에 PHP의 출현은 웹 개발자들에게 그야말로 단비와도 같은 소식이었다. 훗날 젠드(Zend Technologies)를 설립하는 천재 개발자들이 PHP 팀에 합류하고 전체 코드를 다시 쓰고 배포하면서 차츰 언어로서의 면모를 갖추기 시작했다. 당연히 그 인기도 날로 높아졌다.

PHP는 LAMP(Linux, Apache, MySQL, PHP) 스택의 덕을 톡톡히 누렸으며, 웹 호스팅 회사들은 닷컴 열기와 함께 저렴하면서도 안정적인 리눅스 호스팅 상품을 앞다투어 내놓았다. 대안이 많지 않았던 시기에 LAMP 스택이 PHP에 날개를 달아 준 셈이었다.

웹은 닷컴 버블이 꺼진 후, 2000년 중반에 또 한 번의 큰 변곡점을 맞이한다. 우리가 알고 있는 대부분의 웹 프레임워크와 아마존 웹 서비스, 깃허브, 스택 오버플로, 트위터, 페이스북도 모두 이 시기에 등장했다. 당시까지만 해도 전 세계 웹 트래픽의 80%를 PHP가 점유하고 있었는데, 이젠 바야흐로 무한 경쟁 시대에 돌입한 것이다.

마이너 버전 업은 계속 있었지만, PHP 5와 PHP 7 사이의 공백은 자그마치 11년이나 된다. 이 때문인지 PHP 언어, 프레임워크, 생태계는 멈춰 있는 듯했다. 과거의 성공에 심취한 일부 개발자들은 옛날의 코딩 방식에 틀어박혀 세상이 변하는 것을 애써 무시했고, PHP 세계가 잠깐 졸고 있는 사이 다른 언어들도 웹 초창기의 PHP만큼 큰 파도를 일으키면서 웹 분야에 파고들었다. PHP 세계에서도 사랑받는 프레임워크가 있었으나, 다른 언어의 웹 프레임워크에 견줄 만큼 풍부하고 견고하다 할 수는 없었다.

나는 2012년 안드로이드 애플리케이션 개발팀을 맡으면서 프로그래밍의 세계로 다시 돌아왔다. 그때는 루비 온 레일즈로 서버 개발을 시작했는데, 루비 온 레일즈와 비슷한 PHP 웹 프레임워크를 찾다가 라라벨 3를 만났다. 라라벨이 처음 공개되고 불과 일 년이 지났을 시점이었다.

2002년에는 HTML, CSS, 자바스크립트, PHP/ASP, SQL 정도만 알면 웹 개발자로 밥벌이를 할 수 있었다. 하지만 2012년에 웹 개발자로 살기 위해서는 다음과 같은 지식이 필요했다.

- 개발 환경 셋업
- 프로그래밍
  - 서버 사이드
    - 새로운 언어 스펙, 프레임워크
    - 데이터베이스
    - 메시지 큐 등 새로운 개념
  - 클라이언트 사이드
    - Sass, Less
    - CSS 및 자바스크립트 프레임워크
    - 걸프 등 빌드 도구
  - 테스트
  - 버전 관리
- 배포
  - 지속적 통합 및 배포 자동화
- 서버
  - 클라우드 및 분산 아키텍처
  - 프로비전

- 운영 및 문제 해결
- 최적화

사실 라라벨을 처음 접했을 때도 웹 개발에 필요한 대부분의 기능을 담고 있었지만, 버전이 올라가면서 차츰 기능을 보강해 나갔다. 지금 와서 생각해 보면 라라벨을 만난 것은 정말 행운이었다고 생각한다.

책을 저술하고 그간의 PHP 발전사를 쫓아가면서 PHP의 철학과 개발자들의 기대 사이에 엄청난 간극이 존재했고('PHP: 잘못된 디자인의 프랙탈' 문서를 살펴보시라), 그 간극을 라라벨이 훌륭히 메꾸고 있다는 생각을 했다. 예를 들어 초기화되지 않은 변수를 쓰려고 했을 때는 경고만 내고 좋은 게 좋은 것처럼 사용했다면, 라라벨에서는 엄격하게 예외로 처리한다. 하지만 그렇다고 PHP의 장점을 완전히 버린 것은 또 아니다.

입문자뿐만 아니라 중급자에게도 자신 있게 이 책을 권한다. 이 책은 PHP 언어 또는 웹 프로그래밍을 한 번도 접해 보지 않은 독자도 읽을 수 있도록 부록에서 입문 지식을 제공하고 있다. 다른 프레임워크를 사용해 보신 독자라면 라라벨을 더욱 쉽게 배울 수 있으며, 반대로 라라벨을 익힌 사용자는 익스프레스, 레일즈, 장고 등을 학습하는 것이 수월할 것이다. API 서버를 개발하는 실전 프로젝트에서는 HTTP 스펙 등 고급 주제도 다루므로 여러 층의 독자에게 도움이 될 거라 장담한다.

프레임워크 상호 운용성 그룹의 자발적 표준화 활동, 표준 의존성 관리자의 출현, 라라벨 등 걸출한 프레임워크의 성장, PHP 7의 출시 등 지난 몇 년 동안의 행보를 보면 PHP는 아직 죽지 않았으며, 오히려 다시 중흥기를 맞이하고 있다. 어떤 언어나 프레임워크를 선택하든 눈을 부릅뜨고 세상의 흐름을 놓치지 않으면서 내 앞의 문제를 해결하려는 능력이 중요하다. 이 책이 여러분들이 깨어 있도록 하는 데 작으나마 도움이 되기를 바란다.

감사의 글

최근에 라라벨 개발자를 구인한다는 소식을 많이 접한다. 작년과는 온도가 다르다. 전혀 그럴 필요가 없었는데, 오랫동안 라라벨을 쓴 사용자로서 국내 라라벨 확산에 대한 묘한 사명감이 있었다. 책을 쓰고 나서 마음이 한결 편해졌다.

가장 먼저 독자 여러분께 감사드린다. 그리고 모자란 글을 출판해 독자 여러분과 만나게 해준 제이펍에 감사드린다.

거의 반 년간의 집필 기간을 참고 기다려 준 가족들을 생각하면 미안함이 앞선다. 포기하고 싶었던 고난의 시간마다 희망으로 날 이끌어 주어 고맙다.

마지막으로 항상 응원해 주시는 직장 및 커뮤니티 동료들에게 깊이 감사드린다.

지은이 **김주원(appkr)**

베타리더 후기

김용균(이상한 모임)

라라벨에 대해 기초부터 실무에 가까운 코드까지 모두 훑어볼 수 있는 책입니다. 단순히 프레임워크만 설명하는 데 그치지 않고 모던 PHP를 사용하기 위해서 알아야 할 다양한 배경과 지식을 함께 전달하고 있습니다. 꼭 라라벨이 아니더라도 PHP를 사용하고 있다면 이 책을 읽어 실력을 한 단계 끌어올릴 수 있을 것이라 자부합니다. 탁월하고 좋은 내용의 책이었습니다.

김진영(한국정보공학 D&S)

자바를 사용하는 프로그래머지만, 다른 언어와 프레임워크도 알아야 한다는 생각에 이 책을 보게 되었습니다. 예전에 PHP 책 한 권 보았다는 얄팍한 경험에 근거한 자신감으로 시작해서 조금 난관을 겪었지만, 결과적으로 프레임워크와 프로그래밍 언어에 대한 이해도를 높일 수 있는 좋은 경험이었습니다. 개인적으로는 부록을 먼저 읽어 보시고 본 내용을 읽으시길 권합니다.

안정수(카카오)

라라벨을 익히고자 하는 사람이라면 꼭 읽어 봐야 할 책입니다. 라라벨의 구성과 기본 개념, 웹 애플리케이션을 제작할 때 필요한 부분을 실습이 가능한 코드와 함께 하나하나 짚어주는 부분에서 저자의 내공을 느낄 수 있습니다. 이 책을 통해서 모던 PHP를 제대로 맛볼 수 있습니다. 다만 완벽히 초심자들을 대상으로 한 입문용이라고 하기에는 약간 어려울 수 있으므로 어느 정도 지식을 가지신 분이 읽으신다면 더할 나위 없이 좋을 것입니다.

한홍근(lablup)

기존 5.x 버전보다 눈에 띄는 성능 향상을 보인 PHP 7과 PHP를 기반으로 한 웹 개발 프레임워크 '라라벨' 프레임워크를 이용해 하나의 프로젝트를 통해 웹 개발을 해볼 수 있게 도와줍니다. Git을 통한 버전 관리와 AWS에서 결과물을 실제로 작동시켜 봅니다. 웹 개발 입문을 망설이고 있는 분이라면 꼭 읽어보셨으면 좋겠습니다. git을 활용한 점도 좋았습니다.

허찬순(삼성전자)

아무것도 모르는 상태에서 시작하기보다는 웹 프로그래밍뿐만 아니라 프로그래밍 전반(PHP, 자바스크립트, MySQL, git 등)에 대해 넓고 얕은 지식을 갖고 있는 상태에서 책을 읽는 편이 좋습니다. 이 책은 라라벨 프레임워크 기능 전반에 관해 설명하고 있으며, 다양한 용어들이 나와 당황할 수도 있겠지만 심오하거나 난해하지는 않으니 지레 겁먹을 필요는 없습니다. 저자의 설명대로 차근차근 따라 해보길 권합니다.

제이펍은 책에 대한 애정과 기술에 대한 열정이 뜨거운 베타리더들로 하여금
출간되는 모든 서적에 사전 검증을 시행하고 있습니다.

들어가며

I 이 책에 대하여

이 책은 PHP 프로그래밍 언어로 작성된 웹 프레임워크인 라라벨을 다룬다.

라라벨 프레임워크는 2011년에 처음 공개됐다(이하 '라라벨'로 통일). 2013년 버전 4가 공개된 후 PHP 개발자들에게 가장 사랑받는 웹 프레임워크의 자리를 줄곧 지키고 있다.

이 책은 풀 스택(full stack)과 데브옵스(DevOps)를 지향하는 웹 개발자가 알아야 할 폭넓은 내용을 담고 있다. 이는 라라벨이 지향하는 바이기도 하다. 라라벨은 웹 프레임워크뿐만 아니라 개발 환경, 서버 프로비전, 코드 배포까지 웹 서비스 개발에 필요한 모든 도구를 제공한다.

라라벨만 알아서는 웹 서비스를 결코 만들 수 없다. 기획, UI와 사용자 경험, 데이터 모델링, 비즈니스 로직, HTTP 프로토콜, 서버 준비 및 운영 등 웹 프로그래머가 갖추어야 할 소양들도 소개한다. 지금 웹 개발에 입문하려는 독자라면 이 책을 통해 웹 프로그래밍을 위해서 어떤 기술을 더 익혀야 할지 실마리를 얻기 바란다.

목적

이 책은 다음과 같은 목적을 가지고 있다.

- 라라벨 입문 돕기
- 모던 웹 개발 방법론과 모범 사례를 전파
- 실전 프로젝트를 통해 중급 개발자로 성장할 수 있도록 돕기

대상

이 책은 다음 독자들을 대상으로 한다.

- PHP 언어로 웹 개발에 입문하려는 독자
- 라라벨에 입문하려는 독자

나는 PHP 언어로 웹 개발을 시작하려는 사람들에게는 반드시 라라벨로 시작하라고 강력히 권한다. 흔히 입문이 어렵다고 알려져 있지만 그것은 오해다. 다른 PHP 프레임워크 또는 다른 프로그래밍 언어로 웹 개발을 한 경험이 있는 독자라면, 이 책이 라라벨의 매력을 경험해 볼 좋은 기회가 될 것이다.

라라벨은 다른 프로그래밍 언어로 작성된 웹 프레임워크[*1]에 비해 부족함 없이 모든 기능 요소를 포함하고 있다. 따라서 풀 스택 웹 프레임워크라 불린다. 라라벨 공식 문서를 처음 접하는 사용자는 그 완결성뿐만 아니라, 처음 보는 무서운 용어들에 지레 겁을 먹어 마냥 어렵다고만 느낄 수 있다. 그런데 실제로 라라벨에서 쓰이는 용어는 몇 개뿐이며, 그 용어들도 흔히 쓰는 기술 용어에 새로운 이름을 붙였을 뿐이다(예: 블레이드, 엘로퀀트).

라라벨은 입문자부터 고급 사용자까지 다양한 사용자층을 두루 포용한다. 사실 라라벨 입문은 쉬운 편인 데다, 갓 입문해서 배운 지식만으로도 웹 서비스를 개발할 수 있다. 다만, 매사가 그렇듯이 잘 해내기가 어려울 뿐이다(이하 '웹 서비스 개발'은 '웹 개발'로 줄여서 쓰겠다).

PHP 언어와 객체 지향 프로그래밍에 익숙하지 않은 독자는 부록 B와 C를 먼저 읽고 이 책을 살펴볼 것을 권한다.

이 책에서 자세히 설명하지 않는 내용

이 책에서는 다음의 내용에 대해서는 자세히 설명하지 않는다.

- HTML, CSS, 자바스크립트 등 프런트엔드
- 데이터베이스와 데이터베이스 쿼리(라라벨은 클래스 문법으로 데이터베이스를 조작할 수 있다.)

---

*1 다른 프로그래밍 언어로 만든 대표적인 웹 프레임워크로는 루비 언어로 만든 루비 온 레일즈(Ruby on Rails)와 파이썬 언어로 만든 장고(Django)를 들 수 있다.

이 책에 실린 실전 프로젝트에서는 바로 써도 될 만큼 완성도 있는 서비스들을 개발한다. 그 과정에서 HTML, CSS, 자바스크립트 등 프런트엔드 코딩을 하고, 데이터베이스를 설계하고 쿼리하는 작업을 한다. 이 책의 주제는 라라벨이기 때문에 나머지 기술들은 꼭 필요한 경우에만 설명하겠다.

이 책의 구성

이 책은 중급자들이 필요한 부분만 사전식으로 찾아보기 위한 형식의 책이 아니다. 총 5부에 걸쳐 라라벨을 처음 접하는 독자들을 위한 단계별 학습을 제공한다.

1부 라라벨 입문

입문자가 알아야 할 내용만 추렸다.

- 라라벨의 전역 환경 설정법을 익히고, 라우팅과 템플릿을 배운다. 이것만 알아도 간단한 프로젝트를 해낼 수 있다.
- 데이터베이스와 모델, 컨트롤러 등 MVC 구조를 익힌다. 여기까지 익히면 좀 더 나은 구조의 코드를 짤 수 있다.
- 사용자 인증, 메일 보내기, 이벤트, 유효성 검사 등 고급 웹 서비스 구성에 필요한 요소들을 학습한다.

개발 환경 준비와 실습을 포함해서 이틀을 온전히 투자하면 공부할 수 있는 양이다(내가 오프라인 강의를 통해 경험한 사실이다). 여기까지 학습하면 라라벨의 MVC 구조와 기본 문법을 이해하고 이어지는 실전 프로젝트를 진행할 수 있다.

2~4부 실전 프로젝트

- **2부 실전 프로젝트 I — 마크다운 뷰어**
  - PHP의 표준 의존성 관리 도구로 외부 컴포넌트를 가져와 라라벨의 기능을 확장하는 방법을 배운다.
  - 라라벨의 캐시 기능과 프런트엔드 리소스 빌드 자동화 기능 등을 학습한다.

- **3부 실전 프로젝트 II — 포럼**
  - CRUD[*2]를 마스터한다.
  - 클라이언트와 서버 간의 HTTP 통신 원리를 이해한다.
  - 사용자 인증과 인가, 파일 업로드, 소셜 로그인, 다국어 지원, 풀 텍스트 검색, 콘솔 명령 등의 기능을 학습한다.
- **4부 실전 프로젝트 III — RESTful API 서비스**
  - 3부에서 만든 포럼에서 생성된 데이터를 다양한 클라이언트에게 제공하는 서비스를 구현한다.
  - HTTP의 무상태(stateless) 특성과 API 클라이언트 인증 방법을 배운다.
  - 사용량 제한(rate limit)과 캐싱, 보안 등 고급 주제들을 경험해 본다.

5부 코드 배포

우리가 개발한 웹 서비스를 운영(production) 서버에 배포하는 방법을 학습한다.

- 서버를 구축한다.
- 라라벨의 원격 작업 실행 도구 사용법을 배운다.
- 배포 자동화 스크립트를 작성한다.

부록 A 운영체제별 개발 환경 준비

Mac OS X, 우분투 데스크톱(Ubuntu Desktop), Windows 10 운영체제에서 개발 환경을 준비하는 방법을 다룬다.

부록 B 속성 프로그래밍 입문

PHP 언어를 처음 접하는 독자들을 위해 변수와 자료형, 연산자, 제어 구조, 함수, 클로저 등을 소개한다.

부록 C 속성 PHP 객체 지향 프로그래밍 입문

클래스와 클래스 멤버, 캡슐화, 상속, 인터페이스 정도만 소개한다. 부족한 부분들은 본문 중간중간 추가로 설명한다.

---

*2 CRUD(create, read, update, delete) — 데이터베이스의 테이블에 새로운 레코드를 생성하고, 기존 레코드를 조회하고 변경하거나 삭제하는 행위.

II 일러 두기

이 책을 읽는 독자층은 무척 다양할 것이다. 실력 있는 독자분들은 내가 책을 통해 어떤 도구를 권하는지에 개의치 않고 자신이 가진 도구를 통해 자력으로 해석해서 사용할 것이다. 이미 PHP와 개발 도구에 익숙한 독자라면 내 주장대로 라라벨 입문이 그리 어렵지 않다. 반면, 개발 환경조차 꾸미지 못하는 분들도 있다는 것을 이해해 주시면 좋겠다. 이런 분들은 도구 선택의 자유도를 드리는 것보다는 오히려 개발 환경을 강제하는 것이 많은 문제점을 피해가는 방법이라 생각한다.

자신의 환경을 이해하고 제어할 수 있는 독자라면 자신이 선호하는 환경을 사용하시기 바란다(예: PHP 내장 웹 서버 대신 엔진엑스(NGINX), 크롬 대신 사파리).

라라벨 버전

이 책과 소스 코드는 라라벨 프레임워크(laravel/laravel) 5.3.0, 핵심 컴포넌트(laravel/framework) 5.3.9 버전을 기준으로 한다. 라라벨 5.1 버전(LTS[*3]) 또는 5.2 버전과 용법이 다른 부분은 병행해가며 설명하겠다.

라라벨의 업데이트 속도는 상당히 빠른데, 이는 프레임워크가 살아 있다는 증거다. 독자 여러분이 이 책을 읽을 쯤에는 5.3.9와 다른 버전을 설치하여 사용하고 있을 것이다. 라라벨이 6.x가 아니라면 이 책의 모든 코드가 동작한다. 그리고 6.x 시절이 오더라도 이 책에서 학습한 내용은 대부분 유효하다는 것을 보증한다.

용어 혼동 주의

PHP는 웹 개발 분야에 많이 쓰는 '프로그래밍 언어'다. PHP 프로그래밍 언어의 문법에 맞게 쓴 프로그램을 'PHP 스크립트'라 부른다. 또한, 컴퓨터에 설치하여 PHP 스크립트를 해석하고 실행하는 소프트웨어를 'PHP 인터프리터'라 부른다.

---

*3 장기 지원 버전(LTS, Long Term Support) — 라라벨 5.1은 장기 지원 버전이다(2015년 6월에 공개). 라라벨의 장기 지원 정책은 버그 수정 2년, 보안 패치 3년이다.

그런데 우리는 PHP 프로그래밍 언어, PHP 스크립트, PHP 인터프리터를 구분하지 않고 편의상 PHP라 부른다. 이 책에서도 PHP란 단어는 일반적인 용법을 따르므로 문맥에 맞춰 이해하기 바란다.

PHP 내장 웹 서버

이 책에서는 전용 웹 서버(아파치, 엔진엑스, IIS) 대신 PHP 내장 웹 서버를 이용하는 것으로 설명한다. 라라벨 프로젝트 디렉터리에서 ii-1 명령으로 PHP 내장 웹 서버를 실행하면, 브라우저에서 실습 예제를 확인할 수 있다(편의상 '로컬 서버'라 부르기도 한다). 참고로, 전용 웹 서버를 사용하는 독자는 이 명령을 실행할 필요가 없다.

콘솔 ii-1 PHP 내장 웹 서버 실행

```
$ php artisan serve
# Laravel development server started on http://localhost:8000/
```

이제 웹 브라우저에서 http://localhost:8000으로 접속해서 실습 예제의 동작을 확인한다. 다른 호스트 이름이나 포트를 사용하고 싶다면 --host, --port 옵션을 이용하도록 하자.

콘솔 ii-2 PHP 내장 웹 서버의 호스트와 포트

```
$ php artisan serve --host=0.0.0.0 --port=8001
```

0.0.0.0 호스트를 이용하면 로컬 서버와 같은 네트워크에 연결된 다른 컴퓨터에서 해당 서버에 접속할 수 있다. 이때 로컬 서버의 주소는 $ ifconfig(Windows에서는 $ ipconfig) 명령으로 확인한다.

웹 브라우저와 URL

본문에서 브라우저의 개발자 도구를 이용할 때 크롬을 기준으로 설명한다. 본문에 사용한 브라우저 화면 캡처도 모두 크롬이다. 실습을 위해 다른 웹 브라우저를 사용해도 무방하다.

웹 서버의 호스트와 포트는 독자마다 다를 수 있으므로, 본문에서 URL을 표현할 때 호스트:포트를 제외한 경로를 HTTP 메서드와 함께 표시한다. 예를 들어, GET /articles/create는 웹 브라우저 주소 표시줄에 http://호스트:포트/articles/create를 입력한다는 의미다.

콘솔

영화에서 해커들이 오직 키보드 타이핑만으로 컴퓨터와 응용프로그램을 빠르게 제어하는 도구를 보았을 것이다. 이 도구를 콘솔이라 한다. 해커들이 GUI를 사용하지 않고 콘솔에서 명령줄 인터페이스(CLI)를 사용하는 이유는 매우 간단한데, 바로 생산성 때문이다.

라라벨은 다른 PHP 프레임워크에 비해 명령줄 도구를 많이 이용하는 편이다. 반복되는 작업을 수행하거나 뼈대 코드(boilerplate code)를 빠르게 만들 때 주로 사용한다.

운영체제의 콘솔 명령은 $로 시작하고, MySQL 데이터베이스의 콘솔은 mysql>로 시작하며, REPL의 경우에는 >>> 또는 php>로 시작한다. 이런 기호를 **프롬프트(prompt)**라 한다.

책 본문에서 프롬프트를 만나면 콘솔에서 명령을 실행한다. `$ ls`라는 구문은 $를 빼고 ls를 타이핑한 후 Enter 키를 누르라는 의미다. 콘솔 명령 박스에서 #은 주석이므로 콘솔에는 입력하지 않는다.

콘솔 ii-3 디렉터리 목록 조회하기

```
$ ls
```

콘솔 명령을 대체할 수 있는 GUI 툴이 있는 경우, GUI 툴을 사용해도 무방하다. 나는 콘솔과 명령줄 인터페이스에 익숙해질 것을 권장한다. GUI는 시간이 지남에 따라 메뉴의 위치나 이름이 바뀌지만, 콘솔 명령은 바뀌지 않아서 한번 배워두면 계속 사용할 수 있다. 모든 운영체제에서 공통으로 사용할 수 있다는 이점도 있다(Windows 10에 배시 셸을 탑재한다고 발표했다).

코드 블록

이 책에는 예제가 많다. 여러 페이지에 이어지는 완전한 PHP 스크립트를 수록한 후 코드의 내용을 설명하면, 호흡이 끊겨 읽기에 매끄럽지 않을 수 있으므로 작은 코드 블록으로 나누어 설명과 함께 기재한다. 또한 PHP 여는 태그, artisan 명령으로 생성한 코드의 use 구문이나 주석 등 설명과 무관한 코드는 생략하여 집중력이 분산되지 않도록 할 것이다.

예를 들어 다음 코드는 app/Providers/AppServiceProvider.php의 코드 일부를 나타낸다. 중간에 표시된 `// ...`은 기존과 같은 코드를 생략한 것이다.

코드 ii-1 app/Providers/AppServiceProvider.php

```
class AppServiceProvider extends ServiceProvider
{
    public function register(
    {
        $this->app->singleton(Optimus::class, function () {
            return new Optimus(182961291, 1384265379, 30817169);
        });

        // ...
    }
}
```

III 소스 코드 활용

이 책의 전체 소스 코드는 깃허브(github)[*4]에 공개되어 있으므로 언제든 로컬 컴퓨터로 복제하고 사용할 수 있다.

'아는 것과 할 수 있는 것'은 다르다. 복제한 소스 코드는 참고 및 작동 확인을 위해서만 사용하고, 직접 한 줄씩 따라 하며 라라벨을 학습할 것을 권한다.

소스 코드 복제

콘솔 iii-1 명령으로 로컬 컴퓨터에 이 책의 전체 소스 코드를 다운로드할 수 있다. Windows 사용자도 깃 배시(Git Bash)에서 `git` 명령어를 이용할 수 있다.

콘솔 iii-1 소스 코드 복제

```
# 소스 코드를 myapp 디렉터리에 복제한다.
$ git clone git@github.com:appkr/l5code.git myapp

# 깃허브에 SSH 키가 등록되어 있지 않다면 다음 명령을 이용한다.
$ git clone https://github.com/appkr/l5code.git myapp
```

*4 이 책의 소스 코드는 https://github.com/appkr/l5code에서 다운로드할 수 있다.

GUI 환경을 선호하는 독자는 깃허브 데스크톱[*5] 프로그램을 이용할 수 있다.

태그 이동

매번 실습이 끝날 때마다 깃 버전 관리 시스템으로 커밋(commit) 메시지와 태그(tag)를 부여할 것이다. 깃허브의 커밋 메뉴 또는 깃의 비교 도구(diff)를 이용하면 직전 커밋 대비 소스 코드의 변경 사항을 쉽게 확인할 수 있다. 그리고 콘솔 iii-2 명령으로 원하는 위치로 이동할 수 있다.

콘솔 iii-2 1001-installation 태그로 이동하기

```
$ git checkout 1001       # 여기까지 입력한 후 'Tab' 키를 누르고 'Enter'를 누른다.
$ composer dump-autoload # 오토로드 레지스트리 업데이트
```

TIP

명령줄이나 코드 에디터에서 Tab 키는 전체 명령의 일부만 타이핑하고 나머지를 자동 완성할 때 사용한다. 경로, 명령, 변수, 함수 등을 입력할 때 편리하게 사용할 수 있다.

소스 코드 구동하기

복제한 소스 코드의 작동을 확인하려면 다음 설명을 따른다. 무슨 말인지 모르겠다면 따라하지 말고 그냥 넘어가자. 어차피 1부를 다 읽으면 이해할 수 있을 것이다.

프로젝트의 의존성 설치

다운로드한 소스 코드는 이 프로젝트가 의존하는 컴포넌트들을 포함하지 않고 컴포저(composer)로 이 프로젝트가 의존하는 컴포넌트를 설치한다(컴포저가 없다면 부록 A를 참고해서 설치하자).

콘솔 iii-3 의존성 설치

```
# 소스 코드를 복제한 myapp 디렉터리로 이동한다.
$ cd myapp

# (태그로 이동했을 때를 대비해서) 마스터 가지(branch)로 이동한다.
```

*5 깃허브 데스크톱 — https://desktop.github.com

```
$ git checkout master

# 이 프로젝트가 의존하는 컴포넌트를 설치한다.
$ composer install
```

환경 설정

먼저 .env.example 파일을 복사하여 .env 파일을 만든다. 파일을 열어 자신의 환경에 맞게 적절히 수정하고 저장한다. 예를 들어 사용할 데이터베이스가 foo라면 DB_DATABASE=foo로 수정한다.

콘솔 iii-4 프로젝트 전역 환경 설정 파일 만들기

```
$ cd myapp
$ cp .env.example .env
```

암호화 키를 만든다.

콘솔 iii-5 암호화 키 만들기

```
$ cd myapp
$ php artisan key:generate
# 생성된 암호화 키는 .env 파일에 이미 기록되었다.
```

Mac 또는 리눅스(Linux)를 사용한다면, storage, bootstrap/cache, public/files 디렉터리의 권한을 변경한다.

콘솔 iii-6 디렉터리 권한 변경

```
$ chmod -R 775 storage bootstrap/cache public/files
```

데이터베이스 마이그레이션 및 시딩

테이블을 만들고 더미 데이터를 심는다(.env 파일에서 선언한 데이터베이스에 접속할 수 있어야 한다).

콘솔 iii-7 데이터베이스 마이그레이션

```
$ php artisan migrate --seed --force
```

IV 라라벨은?

라라벨의 철학

라라벨의 철학은 다음과 같다.

- 라라벨의 설치-개발-배포 전 과정이 쉽고 즐거워야 한다.
- 매번 반복되는 일은 라라벨이 대신한다.
- 개발자는 핵심 업무에만 집중해야 한다.

웹 개발자의 핵심 업무는 아이디어를 웹 서비스로 만드는 일이다.

라라벨의 성장 과정

테일러 오트웰[*6]은 라라벨 프레임워크를 왜 만들었을까?

새로운 프로젝트를 할 때마다 반복해서 구현해야 하는 기능이 있다. 대표적인 기능이 사용자 모델 및 인증이다. 당시 가장 인기 있던 PHP 웹 프레임워크에는 이 기능이 빠져 있었다. 테일러 오트웰은 이점에 착안하여 회사에서 개발하던 웹 프레임워크를 잘 포장해서 2011년 6월 세상에 첫 버전을 공개했다.

버전 1은 사용자 인증, 클로저를 이용한 라우팅, 엘로퀀트 등의 기능을 담고 있었다. 같은 해 11월에 발표한 두 번째 버전에서는 MVC 구조를 채택했다. 바로 이듬해 2월에 세 번째 버전을 발표했다. 버전 3에서는 아티즌 명령줄 인터페이스와 데이터베이스 마이그레이션 외 많은 기능이 추가되었다. 버전 3 출시 후 다른 PHP 프레임워크에 기여하던 개발자가 라라벨 팀에 대거 합류했다. 나도 이때 라라벨을 처음 접했다.

버전 3는 번들(bundle)이라는 자체적인 확장 관리 도구를 가지고 있었다. 그런데 이즈음 PHP 세계에 큰 사건이 일어났는데, 바로 컴포저(composer)의 등장이다. 라라벨팀은 컴포저를 이용하는 구조로 프레임워크를 완전히 다시 만들었다.

새로운 버전은 2013년 5월에 공개됐다. 버전 4부터 현재의 구조, 즉 `laravel/laravel`과 `laravel/framework`로 분리된 형태를 가지게 되었다. 이 버전에서는 파사드(facade), 데이터베

*6 테일러 오트웰(Taylor Otwell) — http://taylorotwell.com

이스 시딩(database seeding), 큐(queue), 메일(mail) 등 고급 웹 프로그래밍을 위한 기능들이 포함되었다.

다섯 번째 버전은 2015년 2월에 공개됐다. 가장 큰 변화는 네임스페이스와 PSR-4 오토로딩 적용이다. 이제 클래스 이름 충돌 걱정 없이 외부 컴포넌트를 가져와서 프레임워크의 기능을 자유롭게 확장할 수 있게 됐다.

라라벨의 인기

많은 개발자들이 라라벨을 가장 좋아하는 PHP 웹 프레임워크라고 꼽는 이유는 무엇일까?

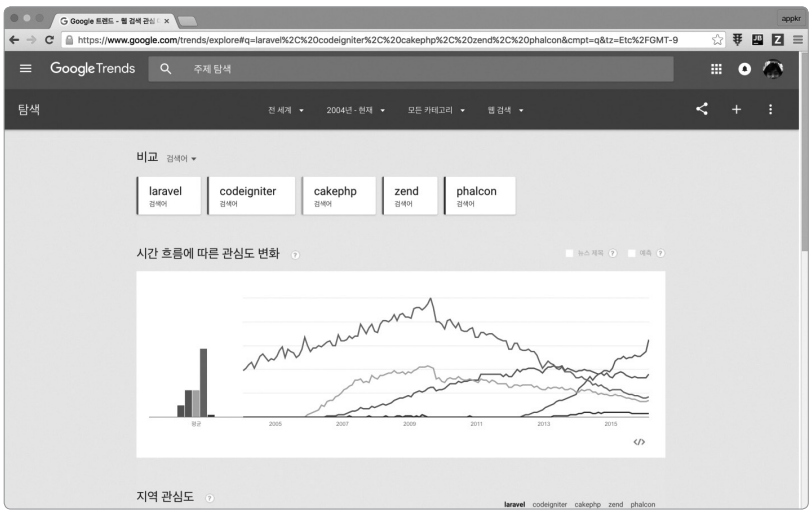


그림 iv-1 구글 트렌드로 본 라라벨의 인기(2016-02-23 기준)

다음 목록은 라라벨 인기 비결에 관한 나의 주장이다.

1. 다른 언어 또는 다른 PHP 웹 프레임워크에서 활동하던 유명 개발자들의 참여[*7]
2. 프레임워크 자체의 우수성
   - 쉽고 고급스러운 문법
   - 미리 만들어 놓은 내장 기능 — 사용자 인증, 캐시 등
   - 강력한 확장 기능들 — 홈스테드(homestead), 소셜라이트(socialite) 등

*7 데일 리즈(Dayle Rees), 그레이엄 캠벨(Graham Campbell), 에릭 반즈(Eric Barnes), 제프리 웨이(Jeffrey Way), 맷 스터퍼(Matt Stauffer) 외

- PSR 표준 준수
- 최신 웹 개발 방법론 적용

3. 라라벨을 이용한 개발 생산성(개발 기간 단축)

라라벨 프로젝트를 한 번만 실행해 보면 이 주장에 동의할 것이다.

커뮤니티와 리소스

라라벨 개발자를 위한 온라인 리소스들이다. 라라벨에 입문하고 나서 고급 개발자로 더 나아가고 싶다면 반드시 살펴보기 바란다.

- 라라벨 뉴스[*8] — 라라벨 관련 뉴스 블로그
- 라라캐스트[*9] — 동영상 강의 서비스. 매주 두세 개의 강의가 올라오며, 수백 편의 기존 동영상 강의를 볼 수 있다($10/월). 게다가 라라벨 커뮤니티에서 가장 활발한 포럼을 운영한다(포럼 참여는 무료)
- LARAVEL.IO[*10] — 라라캐스트가 나오기 전까지 가장 활발하던 포럼

국내에서도 라라벨로 향하는 관심이 서서히 높아지고 있다.

- 우리말로 된 라라벨 5 매뉴얼[*11]
- 라라벨코리아[*12] — 라라벨 팁, 질문과 답변, 구인구직
- 페이스북 그룹[*13] — 라라벨 관련 소식 공유
- 모던 PHP 사용자 그룹[*14]
- PHP THE RIGHT WAY 한글 번역[*15] — 모던 PHP 개발 방법론 학습을 위해 꼭 읽어볼 것을 권장한다.

---

*8 라라벨 뉴스 — https://laravel-news.com

*9 라라캐스트 — https://laracasts.com

*10 LARAVEL.IO — http://laravel.io/forum

*11 라라벨 5 한국어 매뉴얼 — http://laravel.kr/docs

*12 라라벨코리아 — https://www.laravel.co.kr

*13 페이스북 그룹 — https://www.facebook.com/groups/laravelkorea

*14 모던 PHP 사용자 그룹 — http://www.modernpug.org

*15 PHP THE RIGHT WAY — http://modernpug.github.io/php-the-right-way

PART

1

라라벨 입문

웹 프레임워크는 웹 서비스를 개발할 때마다 해야 하는 사용자 인증 및 세션 유지, 데이터베이스 접속, 템플릿 엔진 등을 미리 잘 포장해 놓은 프로젝트의 뼈대 구조다. 프레임워크는 개발자를 대신해서 많은 결정을 내리는 한편, 프레임워크가 제시하는 일련의 관례(또는 문법)를 준수할 것을 요구한다.

이번 1부에서는 MVC 웹 프레임워크의 기본 구성 요소들을 살펴보고, 라라벨이 제공하는 구성 요소들의 사용법을 익힌다. '중요한 부분'과 '쉬운 부분', 그리고 '뒤에서 배우기 위해 먼저 공부해야 하는 부분'을 앞쪽에 배치했다.

CHAPTER

1

라라벨 설치

이번 장에서는 새로운 라라벨 프로젝트를 만들고, 환영 페이지를 띄워 새로 만든 프로젝트가 잘 작동하는지를 확인한다. 그리고 새 프로젝트의 디렉터리 구조와 라라벨의 전반적인 작동 원리를 살펴보겠다.

라라벨 동작에 필요한 개발 환경은 이미 준비되어 있는 것으로 가정한다. 준비되지 않은 독자는 부록 A를 참고해서 준비해 두도록 하자.

1.1 새로운 라라벨 프로젝트 만들기

컴포저로 myapp 디렉터리에 새로운 라라벨 프로젝트를 만든다. 맨 처음으로 설치할 때는 의존성을 계산하고 필요한 컴포넌트를 내려받기 때문에 오래 걸린다.

콘솔 1-1 라라벨 설치

```
# --verbose 옵션은 생략해도 좋다. 설치 과정을 화면에 좀 더 자세히 출력하기 위해 사용하는 옵션이다.
$ composer create-project laravel/laravel myapp --prefer-dist --verbose
```

1.1.1 설치 확인

설치한 라라벨 프로젝트가 잘 동작하는지 확인한다. 콘솔 1-2에서 사용한 명령어 중 아티즌

(artisan)은 라라벨이 제공하는 명령줄 인터페이스다.

콘솔 1-2 라라벨 설치 확인

```
$ cd myapp
$ php artisan --version
# Laravel Framework version 5.3.x
# laravel/laravel의 버전이 아니고 laravel/framework의 버전이다.
```

브라우저에서도 설치가 잘 됐는지를 확인하기 위해 PHP 내장 웹 서버를 시작한다.

콘솔 1-3 로컬 서버 구동

```
$ php artisan serve
# Laravel development server started on http://localhost: 8000/
```

콘솔을 그대로 둔 채 웹 브라우저 주소 표시줄에 http://localhost:8000을 입력하고, 그림 1-1과 같은 환영 페이지가 표시되는지 확인한다(아파치, 엔진엑스, IIS 등의 전용 웹 서버를 이용하는 독자는 각자의 환경에 맞는 URL을 입력한다).

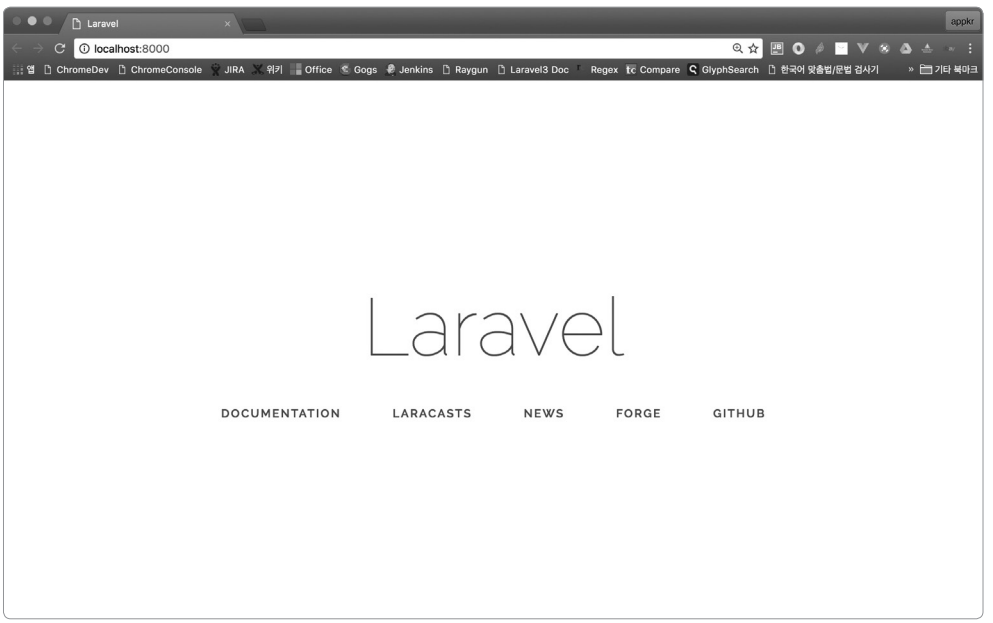


그림 1-1 라라벨 환영 페이지

PHP 내장 웹 서버는 Ctrl + c 키를 눌러서 종료할 수 있다.

TIP

라라벨을 구동하는 로컬 컴퓨터에서 8000번 포트를 이미 사용하고 있다면, PHP 내장 웹 서버를 구동할 때 다른 포트를 사용한다.

```
$ php artisan serve --port=다른_포트_번호
```

1.2 라라벨 프로젝트 구조

각 디렉터리가 하는 일을 간략히 설명한다. 지금 당장 이해가 되지 않는다고 실망하지는 말자. 책을 다 읽고 나서 이 페이지를 다시 펴본다면 분명 모두 이해할 수 있을 것이다.

```
.
├── .env                                  # 글로벌 설정 중 민감한 값,
                                          # 서비스 실행 환경(개발/운영)에 따라 달라져야 하는 값을 써놓은 곳
├── app
│   ├── Console                           # 우리가 만든 콘솔 명령을 담는 디렉터리
│   │   └── Kernel.php                    # 콘솔 명령, 크론 작업을 등록하는 레지스트리
│   ├── Exceptions                        # 우리가 만든 예외(Exception) 클래스를 담는 디렉터리
│   │   └── Handler.php                   # 전역 예외 처리 규칙을 정의한 클래스
│   ├── Http                              # HTTP 요청을 처리하는 클래스들을 담는 디렉터리
│   │   ├── Controllers                   # HTTP 요청을 처리하는 컨트롤러를 담는 디렉터리
│   │   │   └── Auth                      # 라라벨에 기본 내장된 사용자 인증 컨트롤러
│   │   ├── Kernel.php                    # HTTP 요청 처리를 위해 기본이 되는 커널
│   │   └── Middleware                    # 미들웨어를 담는 디렉터리
│   ├── Providers                         # 서비스 프로바이더(service provider)를 담는 디렉터리
│   │   ├── AppServiceProvider.php        # 우리가 만든 서비스를 서비스 컨테이너에 등록하기 위한 클래스
│   │   ├── AuthServiceProvider.php       # 사용자 인가와 관련된 정책을 등록하기 위한 클래스
│   │   ├── BroadcastServiceProvider.php  # 브로드캐스트 메시지 전송을 제어하는 클래스
│   │   ├── EventServiceProvider.php      # 이벤트와 이벤트 처리기를 연결하는 클래스
│   │   └── RouteServiceProvider.php      # routes 디렉터리에서 정의한 라우팅을 활성화하는 클래스
│   └── User.php                          # 기본 내장 User 모델
├── bootstrap                             # 프레임워크 부팅 스크립트
├── composer.json                         # 이 프로젝트의 의존성 및 오토로드 레지스트리
├── composer.lock                         # 현재 환경에 설치한 의존성의 버전 잠금 파일
├── config                                # 데이터베이스, 큐, 메일 등 전역 설정을 담는 디렉터리
├── database
│   ├── factories                         # 더미 모델을 만들기 위한 레시피를 담는 디렉터리
│   ├── migrations                        # 이 프로젝트의 데이터베이스 테이블 스키마를 담는 디렉터리
│   └── seeds                             # 생성한 테이블에 더미 데이터를 삽입하는 레시피를 담는 디렉터리
├── gulpfile.js                           # 엘릭서(elixir, 프런트 엔드 빌드 자동화) 레시피
├── package.json                          # 엘릭서가 의존하는 Node.js 패키지, 이 프로젝트가 의존하는
프런트엔드 리소스 레지스트리
├── phpunit.xml                           # PHPUnit(테스트 프레임워크) 설정
```

```
├── public                  # 웹 서버 루트(document root)
├── resources
│   ├── assets              # 엘릭서 빌드 전의 원본 자바스크립트, CSS 등을 담는 디렉터리
│   ├── lang                # 다국어 지원을 위한 언어별 사전을 담는 디렉터리
│   └── views               # 뷰 파일을 담는 디렉터리
├── routes                  # 라우팅 정의 테이블을 담는 디렉터리(라라벨 5.3에서 신설)
│   ├── api.php             # API 엔드 포인트
│   ├── console.php         # 클로저 형식으로 작성한 아티즌 콘솔 명령
│   └── web.php             # 웹 엔드 포인트
├── server.php              # 로컬 웹 서버 구동을 위한 스크립트
├── storage                 # 라라벨의 파일 저장소(캐시, 로그 등)
├── tests                   # 테스트 파일을 담는 디렉터리
└── vendor                  # 이 프로젝트가 의존하는 PHP 컴포넌트(의존성)를 담는 디렉터리
```

우리가 새로 만든 라라벨 프로젝트는 크게 네 부분으로 나눌 수 있다.

1. 우리가 직접 개발한 애플리케이션 레이어(acme/myapp이라고 하자)
2. laravel/laravel 라라벨 프레임워크. 다음 그림 1-2에서 3번인 '라라벨 핵심 컴포넌트'와 4번인 '외부 컴포넌트'를 잘 조합하여 웹 서비스를 만들 수 있는 기본 틀
3. laravel/framework 라라벨 핵심 컴포넌트(파운데이션 또는 커널이라 부르기도 함)
4. 외부 컴포넌트

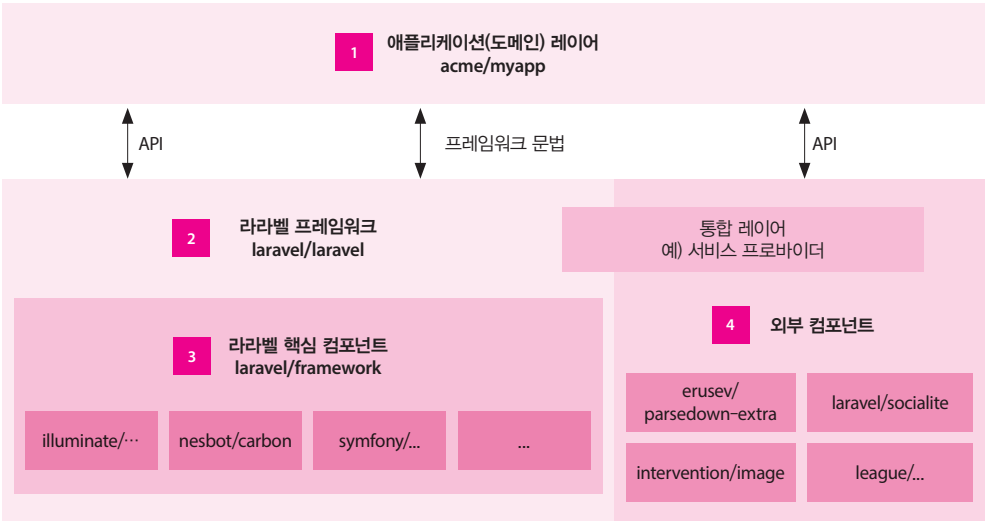


그림 1-2 라라벨 프로젝트 구조

우리가 앞으로 만들 부분이 1번에 해당한다. 콘솔 1-1의 명령은 2번을 설치한 것이다. 새로 만든 프로젝트의 composer.json 파일을 열어 보면 패키지를 설치할 때 3번과 4번을 요구한

다는 내용이 선언되어 있다. 다음과 같은 composer.json을 사용해서 설치를 수행하면 최종적으로 앞서 나온 그림 1-2의 2, 3, 4에 해당하는 내용을 설치하게 된다.

코드 1-1 composer.json

```
{
  "require": {
    "php": ">=5.6.4",
    "laravel/framework": "5.3.*"
  },
  "require-dev": {
    "fzaninotto/faker": "~1.4",
    // "...",
  }
}
```

왜 2와 3으로 나누어 놓았을까? 나도 한동안 이 점을 이해하지 못했다. 이 구조의 진가는 라라벨 프레임워크의 부 버전이 올라갈 때 체험할 수 있다. 라라벨 프레임워크(2)는 우리가 작성한 코드와 엉켜 있는 반면에, 라라벨 핵심 컴포넌트(3)는 우리의 코드와 완전히 분리되어 있다. 버전 변경의 충격을 흡수해 주는 것이다. 정리하자면, 우리가 작성한 코드가 속한 완충 레이어(2)만 지침에 맞도록 업그레이드하면 된다. 이 구조 때문에 새로운 주 버전이 배포되더라도 수정할 코드량이 조금 많아질 뿐, 업그레이드를 아주 못하는 것은 아니다.

NOTE

용어 혼동 주의

앞서 laravel/framework는 이름에 프레임워크란 단어가 있음에도 불구하고 '핵심 컴포넌트'라 불렀다. 이것은 라라벨 팀의 작명 실수인데, 지금은 이를 바로잡을 수 없어서 그냥 쓰고 있다. 이 패키지의 깃허브 페이지를 살펴보면 'Laravel Framework'란 제목 뒤에 '(Kernel)'이라고 조심스럽게 쓰여 있다.

1.3 라라벨 작동 원리

설명을 위해 서버의 이름을 example.com이라 하자.

1. 사용자는 브라우저에서 http://example.com/about 페이지 요청을 한다.
2. 사용자의 요청은 example.com이라는 이름을 가진 서버에 도착한다.
3. 웹 브라우저가 HTTP 프로토콜로 요청했으므로 example.com 서버의 웹 서버가 요청을 처리한다.
   ① 웹 서버는 URL을 해석하고 자신이 해결할 수 있는 파일이면, 파일을 읽어서 곧바로 응답한다(CSS, 이미지 등)
   ② 자신이 해결할 수 없는 파일이면 웹 서버 설정에 따라 작업을 PHP에게 넘긴다.
   ③ PHP에게 작업을 넘길 때는 index.php를 향하도록 URL 경로를 변경하여 넘긴다.
4. index.php에는 라라벨의 부팅 시퀀스가 담겨 있다.
   ① 라라벨은 routes/web.php에 정의한 라우팅 테이블에서 about을 찾는다(**라우팅**).
   ② 일치하는 라우트가 없다면 웹 서버에게 적절한 오류 응답을 반환한다(**전역 예외 처리기**)
   ③ 일치하는 라우트가 있다면 전역 미들웨어와 about 라우트에 정의한 라우트 미들웨어가 HTTP 요청을 필터링한다(**미들웨어**).
   ④ 미들웨어를 통과하지 못하면 예외가 발생한다. 전역 예외 처리기는 웹 서버에게 적절한 HTTP 응답을 반환한다.
   ⑤ 미들웨어를 통과하면 비로소 about 요청을 처리할 컨트롤러에게 작업이 도달한다(**컨트롤러**).
   ⑥ 컨트롤러는 HTTP 요청을 처리한다. 이때 라라벨 컴포넌트, 외부 컴포넌트의 기능, 우리가 만든 기능 등을 이용한다. 처리가 끝나면 HTTP 응답을 만들고 반환한다.
   ⑦ 컨트롤러는 요청을 처리하는 과정에 데이터베이스와 통신을 하기도 한다(**엘로퀀트**).
   ⑧ 컨트롤러는 웹 서버에게 돌려줄 HTTP 응답 본문을 만들 때 템플릿 엔진을 이용하기도 한다(**블레이드**).
5. 웹 서버는 PHP/라라벨 측으로부터 넘겨받은 HTTP 응답을 브라우저에 돌려준다.

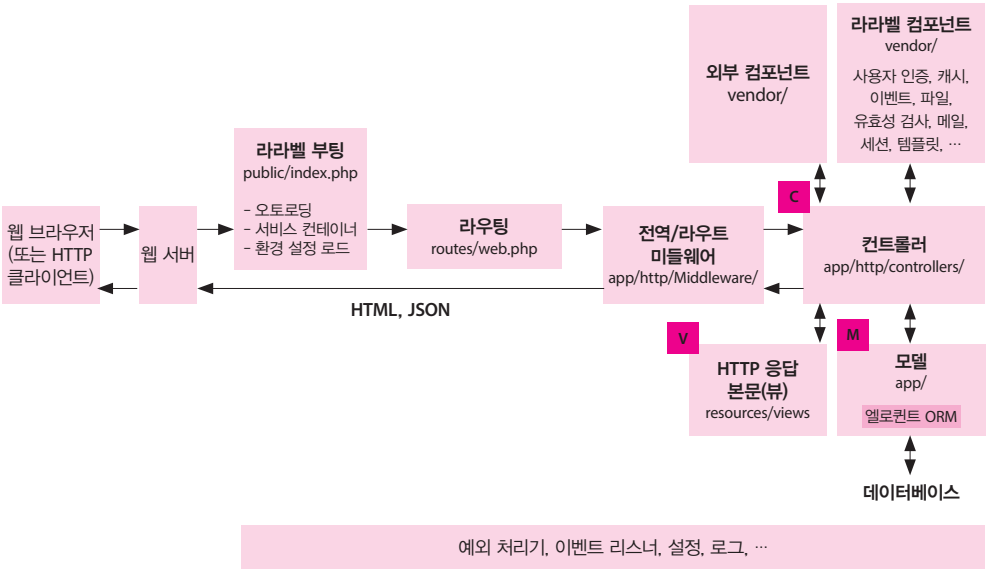


그림 1-3 라라벨로 만든 웹 서비스의 작동 원리

TIP

깃허브 등에서 라라벨로 개발된 소스 코드를 받았다면 가장 먼저 봐야 할 파일은 `composer.json`과 `routes/web.php`다. 어떤 외부 컴포넌트를 이용했는지와 URL과 컨트롤러 간의 연결을 파악하기 위해서다.

1.4 버전 관리

새로운 깃 저장소를 만들고 지금까지의 변경 내용을 기록해 놓자.

콘솔 1-4 깃을 이용한 버전 관리

```
$ git init
# Initialized empty Git repository in /path/to/myapp.git
$ git add .
$ git commit -m '새로운 라라벨 프로젝트 생성'
# [master (root-commit) 3c76f7c] 새로운 라라벨 프로젝트 생성
# 78 files changed, 5580 insertions(+)
# ...
$ git tag 1001-installation
```

CHAPTER

2

전역 환경 설정

웹 서버가 public/index.php를 호출하면 라라벨은 살아나며, HTTP 응답을 웹 서버로 넘겨주면 생을 마감한다. 라라벨의 수명은 대략 수십~수백 밀리 초다.

라라벨이 살아 있는 동안 참조하는 전역 환경 설정은 config 디렉터리 아래에 있다. auth.php, database.php, mail.php 등 이름만 봐도 설정의 대상이 무엇인지 충분히 예측할 수 있는 파일들이다. 파일을 열어 보면, 각 지시자마다 상세한 설명이 달려 있다.

2.1 dotenv 파일이 하는 일

새로 만든 프로젝트의 .gitignore 파일을 열자(.gitignore는 깃 버전 관리에서 제외할 파일과 디렉터리를 기록해 두는 파일이다).

코드 2-1 .gitignore

```
/vendor                 # PHP 컴포넌트
/node_modules           # Node.js 컴포넌트
/public/storage         # 라라벨 파일 저장소(캐시, 로그 등)
# ...
.env                    # '민감한' 전역 환경 변수
```

.gitignore 파일의 마지막 줄에 .env가 보인다. 해석하면 .env 파일은 버전 관리 시스템에 등록하지 않지만, config 디렉터리는 등록한다는 뜻이다. '등록'은 '공유'의 의미로 해석해도 좋다.

config 디렉터리의 파일들과 .env 파일은 어떤 관계일까? .env 파일과 database.php 파일을 번갈아 열며 답을 찾아보자.

코드 2-2 .env

```
# ...
DB_HOST=127.0.0.1          # 데이터베이스 서버 주소
DB_DATABASE=homestead      # 이 프로젝트가 사용할 데이터베이스 이름
DB_USERNAME=homestead      # 데이터베이스에 접속하기 위한 사용자 이름
DB_PASSWORD=secret         # 사용자의 비밀번호
```

코드 2-3 config/database.php

```
return [
    // ...

    'connections' => [
        'mysql' => [
            'driver'        =>      'mysql',
            'host'          =>      env('DB_HOST', 'localhost'),
            'database'      =>      env('DB_DATABASE', 'forge'),
            'username'      =>      env('DB_USERNAME', 'forge'),
            'password'      =>      env('DB_PASSWORD', ''),
            // ...
        ],
];
```

NOTE

dotenv 파일 생성

.env 파일이 없다면 당황하지 말고 다음 명령으로 만들어 보자.

```
# .env.example 파일을 .env 파일로 복사한다.
$ cp .env.example .env
```

database.php는 데이터베이스 접속과 관련된 정보를 담고 있다. 이 파일은 설정값을 하드코드로 대입하지 않고 .env라는 파일에서 읽어 온다. env(string $key, mixed $default) 라

라벨 내장 도우미 함수(이하 '도우미 함수')는 .env 파일에서 $key를 찾아 값을 반환하는 일을 한다.

왜 이렇게 할까? 다음 두 가지 조건에 해당한다면 config 디렉터리 아래에 파일에 직접 쓰는 것 보다 .env를 쓰는 것이 더 낫다.

- 웹 서비스의 실행 환경에 따라 다른 설정값을 적용해야 할 때
- 비밀번호처럼 민감한 정보라 버전 관리 시스템에 등록하면 안 되는 정보일 때

위 내용을 잘 이해했다면, config/database.php 파일에 'password' => 'secret',처럼 직접 써도 된다는 것을 눈치챘을 것이다.
.env에 등록된 내용은 $_SERVER PHP 전역 변수에도 등록된다.

2.2 APP 환경 설정

.env에서 APP_으로 시작하는 부분만 살펴보자. 이 설정들은 config/app.php에서 참조하는 값들이다.

- APP_ENV=local
  애플리케이션(라라벨) 실행 환경을 선언한다. 현재 값인 local은 임의의 값이다. 통상적으로 local, staging, testing, production 등의 값을 사용한다. local은 로컬 컴퓨터, production은 운영 서버에서 사용한다.

- APP_DEBUG=true
  디버그 옵션 활성화 여부를 정의한다. true는 로컬 환경에서만 사용해야 한다. true로 설정하면 예외에 관한 상세한 역추적 로그가 화면에 출력된다. 운영 환경에서 이 내용이 표시되면 해커의 공격 대상이 될 수 있으니 주의하자.

- APP_KEY=base64:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  프레임워크 전반에 걸쳐 암호화 알고리즘의 키 값으로 사용되는 값이다. 예를 들면 브라우저와 주고받는 암호화된 쿠키를 만들거나 해독할 때 사용한다.
  SomeRandomString으로 값이 채워져 있거나 비어 있다면, 다음 명령으로 새로운 암호

화 키를 만든다. 또한, 다른 실행 환경에 코드를 처음 배포했을 때도 다음 명령으로 암호화 키를 만들 수 있다.

```
$ php artisan key:generate
```

- APP_URL=http://localhost
  콘솔에는 $_SERVER['HTTP_HOST'] 값이 없다. 아티즌 명령줄 인터페이스는 이 설정을 대체 값(fallback)으로 사용한다.

NOTE

애플리케이션 키

APP_KEY 값은 신중히 다뤄야 한다. 운영 서버에서는 이 값을 함부로 바꾸면 안 된다.

사용자가 우리 서비스에 접속하면 브라우저에 암호화된 쿠키가 저장된다. APP_KEY 값을 변경하면 접속했던 사용자의 쿠키를 복호화할 수 없어 오류가 발생한다. 개발 중에 이 값을 바꿔 가며 실험해 보면, Illuminate\Contracts\Encryption\DecryptException을 볼 수 있다. 개발 중에는 브라우저의 쿠키를 지우면 쉽게 해결되지만, 이미 사용자 브라우저가 우리 서비스의 쿠키를 가지고 있다면 일일이 찾아내서 지우라고는 할 수 없다.

2.3 마치며

라라벨은 민감한 설정 정보를 은닉하거나 애플리케이션 동작에 필요한 전역 환경 변수를 쉽게 변경할 수 있는 방법을 제공한다. config 디렉터리 아래에 연관 배열(associative array)을 반환하는 파일을 만들면 애플리케이션 어디서든 config('파일.키');로 설정값을 읽을 수 있다.