# 1부 라라벨 입문 (ATX HEADER 1)

## 1장 라라벨 설치 (ATX HEADER 2)

### 1.1. 새로운 라라벨 프로젝트 만들기 (ATX HEADER 3)

#### 1.1.1 설치 확인 (ATX HEADER 4)

##### ATX HEADER 5

###### ATX HEADER 6

# 콘솔 18-2 Documentation 모델 뼈대 코드 만들기   
  
$ php artisan make:model Documentation

// 코드 18-2 app/Http/routes.php  
  
Route::get('docs/{file?}', function($file = null) {  
 $text = (new App\Documentation)->get($file);  
   
 return app(ParsedownExtra::class)->text($text);  
});

<!-- // 코드 19-5 resources/views/docs/show.blade.php -->  
  
@extends('layouts.app')  
  
@section('content')  
 <div class="container">  
 <header class="page-header">  
 <h2>마크다운 뷰어</h2>  
 </header>  
  
 <div class="row">  
 <div class="col-md-3 docs\_\_sidebar">  
 <aside>  
 {!! $index !!}  
 </aside>  
 </div>  
  
 <div class="col-md-9 docs\_\_content">  
 <article>  
 {!! $content !!}  
 </article>  
 </div>  
 </div>  
 </div>  
@stop

/\* 코드 21-2 resources/assets/sass/app.scss \*/  
   
@import '../../../node\_modules/bootstrap-sass/assets/stylesheets/bootstrap';  
@import '../../../node\_modules/font-awesome/scss/font-awesome';  
  
body {  
 background-color: #EEE;  
}

**네임스페이스와 오토로딩**

마이그레이션 클래스는 전역 네임스페이스(\)를 사용한다. 곧 배울 데이더베이스 시딩, 모델 팩터리도 전역이다. 이 녀석들은 모두 database 디렉터리 아래에 위치한다.

라라벨은 PSR-4 오토로딩(autoloading) 표준을 이용한다.

우리는 하나의 파일에 만 줄짜리 코드를 구겨 넣지는 않는다. 보통은 여러 개의 파일로 나누어 코드를 구조화한다. 그래서 하나의 파일에 있는 프로그램 코드가 작동하기 위해서는 다른 파일에 있는 코드를 불러와야 한다. PHP 프로그래밍에서는 require 키워드를 이용해서 다른 파일을 불러왔었다. 그런데 불러올 파일이 많아지면 require 구문을 여러 번 사용해야 하는 불편함이 있다.

오토로딩은 이런 불편함을 해결해 준다. PSR-4는 PHP-FIG에서 제시한 표준화된 클래스 오토로딩 방법이다. 우리 프로젝트의 의존성 레지스트리를 열어보자.

// composer.json  
  
"autoload": {  
 "classmap": [  
 "database"  
 ],  
 "psr-4": {  
 "App\\": "app/"  
 }  
},

App 네임스페이스의 루트는 app 디렉터리라고 선언한 것이다. 그리고 app 디렉터리 아래의 파일들은 PSR-4 방식으로 불러온다고 컴포저에게 말한 것이다. app 디렉터리 아래의 파일들을 열어보면, 파일의 시작 부분은 항상 namespace App\...으로 시작하는 것을 확인할 수 있다.

반면, database 디렉터리는 classmap으로 읽어드리고 있다. 그리고 database 아래에 있는 클래스들은 namespace 선언이 없다. 전역 네임스페이스를 이용한다는 의미다.

요약하면, 코드에서 \App\Author를 언급하면 오토로더는 app/Author.php 파일에 필요한 클래스가 있다는 것을 안다. 그리고 파일의 내용을 미리 읽어 와서 준비해 두는 것이다.

PHP 네임스페이스는 운영체제의 디렉터리를 이용하는 것과 같은 방식으로 이용한다. 디렉터리의 슬래시(/) 구분자 대신 역슬래시(\)를 이용한다(Windows에서 역슬래시는 우리나라 통화기호로 표시된 키로 Enter 키 위에 있다).

전체 클래스 경로(FQCN, Fully Qualified Class Name)는 \App\Author처럼 쓴다. 가령 같은 네임스페이스에 있는 App\Post 클래스에서는 (같은 디렉터리기 때문에) Author와 같이 참조할 수 있다. 반면 App\Http\Controllers\PostController 클래스에서는 \App\Author와 같이 전체 클래 경로를 써야 한다. 마찬가지로 전역(\) 네임스페이스에 존재하는 클래스에서 다른 클래스를 참조할 때는 \로 시작할 필요는 없다. 같은 디렉터리에 있는 파일이라 생각하면 편하다.

참고로, 마이그레이션 또는 롤백 실행 중 ClassNotFound 예외가 발생하면, $ composer dump-autoload --optimize 명령으로 해결한다(--optimize 옵션은 생략 가능). 이 문제는 아티즌 콘솔의 변경 사항을 컴포저가 알지 못하기 때문이다. 컴포저가 만들어 놓은 오토로드 레지스트리를 다시 만들면 해결된다.

This is normal text