

Laravel 빠른 참조

Artisan, 라우팅, Eloquent, Blade, 미들웨어, 인증

Artisan

주요 명령어	
<code>php artisan serve</code>	개발 서버 시작
<code>php artisan make:model Name -m</code>	마이그레이션과 함께 모델 생성
<code>php artisan make:controller NameController</code>	컨트롤러 클래스 생성
<code>php artisan make:middleware Name</code>	미들웨어 클래스 생성
<code>php artisan migrate</code>	대기 중인 마이그레이션 실행
<code>php artisan migrate:rollback</code>	마지막 마이그레이션 배치 롤백
<code>php artisan db:seed</code>	데이터베이스 시드 실행
<code>php artisan tinker</code>	애플리케이션 대화형 REPL
<code>php artisan route:list</code>	등록된 모든 라우트 목록
<code>php artisan cache:clear</code>	애플리케이션 캐시 지우기
<code>php artisan config:clear</code>	캐시된 설정 지우기
<code>php artisan queue:work</code>	큐에 쌓인 잡 처리 시작

라우팅

기본 라우트	
<code>Route::get('/users', [UserController::class, 'index']);</code>	
<code>Route::post('/users', [UserController::class, 'store']);</code>	
<code>Route::put('/users/{id}', [UserController::class, 'update']);</code>	
<code>Route::delete('/users/{id}', [UserController::class, 'destroy']);</code>	

라우트 파라미터 및 그룹

<code>Route::get('/user/{id}', function (int \$id) { return User::findOrFail(\$id); });</code>	
<code>Route::prefix('api')->middleware('auth')->group(function () { Route::get('/profile', [ProfileController::class, 'show']); });</code>	

라우트 기능

<code>->name('route.name')</code>	URL 생성을 위한 이름 있는 라우트
<code>->where('id', '[0-9]+')</code>	파라미터에 정규식 제약
<code>Route::resource()</code>	RESTful 리소스 라우트 (7개)
<code>Route::apiResource()</code>	API 리소스 (create/edit 뷰 없음)
<code>Route::fallback()</code>	매칭되지 않는 라우트의 catch-all

컨트롤러

리소스 컨트롤러	
<code>class PostController extends Controller { public function index() { return view('posts.index', ['posts' => Post::all()]); } }</code>	
<code>public function store(Request \$request) { \$validated = \$request->validate(['title' => 'required max:255']); Post::create(\$validated); return redirect()->route('posts.index'); }</code>	

리소스 메서드

<code>index()</code>	GET /resource -- 전체 목록
<code>create()</code>	GET /resource/create -- 폼 표시
<code>store()</code>	POST /resource -- 새 항목 저장
<code>show(\$id)</code>	GET /resource/{id} -- 단일 표시
<code>edit(\$id)</code>	GET /resource/{id}/edit -- 편집 폼
<code>update(\$id)</code>	PUT /resource/{id} -- 업데이트
<code>destroy(\$id)</code>	DELETE /resource/{id} -- 삭제

Blade 템플릿

레이아웃 및 섹션	
<code>{{!-- layouts/app.blade.php --}}</code>	
<code><html><body> @yield('content') </body></html></code>	
<code>{{!-- pages/home.blade.php --}}</code>	
<code>@extends('layouts.app')</code>	
<code>@section('content')</code>	
<code><h1>Home</h1></code>	
<code>@endsection</code>	

디렉티브

<code>{{ \$var }}</code>	HTML 이스케이프 후 출력
<code>{!! \$html !!}</code>	이스케이프 없이 원본 출력
<code>@if / @elseif / @else</code>	조건 블록
<code>@foreach (\$items as \$item)</code>	컬렉션 반복
<code>@forelse / @empty</code>	비어있을 때 대체 블록이 있는 반복
<code>@include('partial')</code>	다른 Blade 뷰 포함
<code>@component / @slot</code>	재사용 가능한 Blade 컴포넌트
<code>@csrf</code>	CSRF 토큰 히든 필드
<code>@auth / @guest</code>	인증 상태 확인
<code>@error('field')</code>	유효성 검사 오류 표시

Eloquent ORM

모델 기본	
<code>class Post extends Model { protected \$fillable = ['title', 'body', 'user_id']; }</code>	
<code>public function user() { return \$this->belongsTo(User::class); }</code>	

쿼리

<code>Post::all();</code>	<code>// all records</code>
<code>Post::find(1);</code>	<code>// by primary key</code>
<code>Post::where('status', 'published')->get();</code>	
<code>Post::where('views', '>', 100)->orderBy('created_at', 'desc')->first();</code>	

CRUD 작업

<code>\$post = Post::create(['title' => 'New', 'body' => '...']);</code>	
<code>\$post->update(['title' => 'Updated']);</code>	
<code>\$post->delete();</code>	
<code>Post::destroy([1, 2, 3]); // delete by IDs</code>	

관계

<code>hasOne</code>	일대일 (User -> Phone)
<code>hasMany</code>	일대다 (Post -> Comments)
<code>belongsTo</code>	hasOne/hasMany의 역방향
<code>belongsToMany</code>	피벗 테이블을 이용한 다대다
<code>hasManyThrough</code>	중간 모델을 통한 has-many

마이그레이션

테이블 생성	
<code>Schema::create('posts', function (Blueprint \$table) { \$table->id(); \$table->foreignId('user_id')->constrained()->cascadeOnDelete(); \$table->string('title'); \$table->text('body')->nullable(); \$table->timestamps(); });</code>	

컬럼 타입

<code>\$table->id()</code>	자동 증가 BIGINT 기본 키
<code>\$table->string('col', 100)</code>	선택적 길이를 가진 VARCHAR
<code>\$table->text('col')</code>	TEXT 컬럼
<code>\$table->integer('col')</code>	INTEGER 컬럼
<code>\$table->boolean('col')</code>	BOOLEAN 컬럼
<code>\$table->json('col')</code>	JSON 컬럼
<code>\$table->timestamp('col')</code>	TIMESTAMP 컬럼
<code>\$table->timestamps()</code>	created_at 및 updated_at
<code>\$table->softDeletes()</code>	소프트 삭제용 deleted_at

미들웨어

커스텀 미들웨어	
<code>class EnsureAdmin { public function handle(Request \$request, Closure \$next) { if (!\$request->user()?->is_admin) { abort(403); } return \$next(\$request); } }</code>	

등록 및 사용

<code>// bootstrap/app.php</code>	
<code>->withMiddleware(function (Middleware \$middleware) { \$middleware->alias(['admin' => EnsureAdmin::class]); })</code>	
<code>// In routes</code>	
<code>Route::get('/admin', fn() => '...')->middleware('admin');</code>	

내장 미들웨어

<code>auth</code>	인증 필요
<code>guest</code>	인증된 경우 리다이렉트
<code>throttle:60,1</code>	요청 속도 제한 (분당 60회)
<code>verified</code>	이메일 인증 필요
<code>signed</code>	서명된 URL 유효성 검사

인증

Auth 헬퍼	
<code>Auth::check();</code>	<code>// is user logged in?</code>
<code>Auth::user();</code>	<code>// current User model</code>
<code>Auth::id();</code>	<code>// current user ID</code>
<code>Auth::attempt(['email' => \$e, 'password' => \$p]);</code>	
<code>Auth::logout();</code>	

스타터 키트

<code>Laravel Breeze</code>	최소한의 인증 스킴 (Blade 또는 Inertia)
<code>Laravel Jetstream</code>	풀 기능 (팀, 2FA, API 토큰)
<code>Sanctum</code>	SPA / 모바일 API 토큰 인증
<code>Passport</code>	완전한 OAuth2 서버 구현

Laravel 빠른 참조

라우트 보호

```
Route::middleware('auth')->group(function () {
    Route::get('/dashboard', [DashController::class, 'index']);
});
```

유효성 검사

컨트롤러 유효성 검사

```
$validated = $request->validate([
    'title' => 'required|string|max:255',
    'email' => 'required|email|unique:users',
    'age' => 'nullable|integer|min:0',
]);
```

Form Request

```
class StorePostRequest extends FormRequest {
    public function rules(): array {
        return [
            'title' => 'required|max:255',
            'body' => 'required|min:10',
        ];
    }
}
```

주요 규칙

required	필드가 있고 비어있지 않아야 함
string integer boolean	타입 유효성 검사
min:N max:N	최소/최대 길이 또는 값
email	유효한 이메일 형식
unique:table,column	DB 테이블에서 고유해야 함
exists:table,column	DB 테이블에 존재해야 함
in:a,b,c	나열된 값 중 하나여야 함
confirmed	_confirmation 필드와 일치해야 함
date after:date	날짜 유효성 검사

주요 패턴

API 응답

```
return response()->json(['data' => $users], 200);
return response()->json(['error' => 'Not found'], 404);
```

환경 변수 및 설정

```
env('APP_KEY'); // read .env value
config('app.name'); // read config value
config(['app.debug' => true]); // set at runtime
```

유용한 헬퍼

route('name', \$params)	이름 있는 라우트의 URL 생성
redirect()->route('name')	이름 있는 라우트로 리다이렉트
back()->withErrors()	유효성 오류와 함께 뒤로 리다이렉트
abort(404)	HTTP 예외 던지기
collect(\$array)	배열로 Collection 생성
now()	현재 Carbon 날짜시간
cache()->remember()	TTL로 값을 캐시