Laravel 빠른 참조

Artisan, 라우팅, Eloquent, Blade, 미들웨어, 인증

Artisan

주요 명령어

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

라우팅

기본 라우트

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

라우트 파라미터 및 그룹

```
Route::get('/user/{id}', function (int $id) {
    return User::findOrFail($id);
});

Route::prefix('api')->middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'show']);
});
```

라우트 기능

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

컨트롤러

리소스 컨트롤러

```
class PostController extends Controller {
    public function index() {
        return view('posts.index', ['posts' => Post::all()]);
    }

    public function store(Request $request) {
        $validated = $request->validate(['title' => 'required|
max:255']);
        Post::create($validated);
        return redirect()->route('posts.index');
    }
}
```

리소스 메서드

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

Blade 템플릿

레이아웃 및 섹션

```
{{-- layouts/app.blade.php --}}
<html><body>
  @yield('content')
</body></html>

{{-- pages/home.blade.php --}}
@extends('layouts.app')
@section('content')
  <h1>Home</h1>
@endsection
```

디렉티브

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

Eloquent ORM

모델 기본

```
class Post extends Model {
    protected $fillable = ['title', 'body', 'user_id'];

    public function user() {
        return $this->belongsTo(User::class);
    }
}
```

쿼리

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

CRUD 작업

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

관계

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

마이그레이션

테이블 생성

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

컬럼 타입

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

미들웨어

커스텀 미들웨어

```
class EnsureAdmin {
    public function handle(Request $request, Closure $next) {
        if (! $request->user()?->is_admin) {
            abort(403);
        }
        return $next($request);
    }
}
```

등록 및 사용

```
// bootstrap/app.php
->withMiddleware(function (Middleware $middleware) {
    $middleware->alias(['admin' => EnsureAdmin::class]);
})

// In routes
Route::get('/admin', fn() => '...')->middleware('admin');
```

내장 미들웨어

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

인증

Auth 헬퍼

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

스타터 킷

`Laravel Breeze`	최소한의 인증 스캐폴딩 (Blade 또는 Inertia)
`Laravel Jetstream`	풀 기능 (팀, 2FA, API 토큰)
`Sanctum`	SPA / 모바일 API 토큰 인증
`Passport`	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