LARAVEL クイックリファレンス

Artisan、ルーティング、Eloquent、Blade、ミドルウェア、認証

Artisan

よく使うコマンド

`php artisan serve`	開発サーバーを起動
`php artisan make:model Name -m`	マイグレーション付きでモデルを作成
`php artisan make:controller NameController`	コントローラークラスを作成
`php artisan make:middleware Name`	ミドルウェアクラスを作成
`php artisan migrate`	未実行のマイグレーションを実行
`php artisan migrate:rollback`	最後のマイグレーションバッチをロールバック
`php artisan db:seed`	データベースシーダーを実行
`php artisan tinker`	アプリのインタラクティブ REPL
`php artisan route:list`	登録済みルートをすべて表示
`php artisan cache:clear`	アプリケーションキャッシュをクリア
`php artisan config:clear`	キャッシュされた設定をクリア
`php artisan queue:work`	キューに入ったジョブの処理を開始

ルーティング

基本ルート

```
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']);
```

ルートパラメーターとグループ

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

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

ルート機能

`->name('route.name')`	URL 生成のための名前付きルート
`->where('id', '[0-9]+')`	パラメーターへの正規表現制約
`Route::resource()`	RESTful リソースルート（7 ルート）
`Route::apiResource()`	API リソース（create / edit ビューなし）
`Route::fallback()`	マッチしないルートのキャッチオール

コントローラー

リソースコントローラー

```
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');
    }
}
```

リソースメソッド

`index()`	GET /resource -- すべて一覧表示
`create()`	GET /resource/create -- フォームを表示
`store()`	POST /resource -- 新規保存
`show($id)`	GET /resource/{id} -- 1 件表示
`edit($id)`	GET /resource/{id}/edit -- 編集フォーム
`update($id)`	PUT /resource/{id} -- 更新
`destroy($id)`	DELETE /resource/{id} -- 削除

Blade テンプレート

レイアウトとセクション

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

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

ディレクティブ

`{{ $var }}`	HTML エスケープして出力
`{!! $html !!}`	生の（エスケープなし）出力
`@if / @elseif / @else`	条件ブロック
`@foreach ($items as $item)`	コレクションをループ
`@forelse / @empty`	空のフォールバック付きループ
`@include('partial')`	別の Blade ビューをインクルード
`@component / @slot`	再利用可能な Blade コンポーネント
`@csrf`	CSRF トークン隠しフィールド
`@auth / @guest`	認証状態を確認
`@error('field')`	バリデーションエラーを表示

Eloquent ORM

モデルの基本

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

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

クエリ

```
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 操作

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

リレーション

`hasOne`	1 対 1（User -> Phone）
`hasMany`	1 対多（Post -> Comments）
`belongsTo`	hasOne / hasMany の逆
`belongsToMany`	中間テーブルを使った多対多
`hasManyThrough`	中間モデルを経由した has-many

マイグレーション

テーブルの作成

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

カラム型

`$table->id()`	自動インクリメントの BIGINT 主キー
`$table->string('col', 100)`	任意の長さの VARCHAR
`$table->text('col')`	TEXT カラム
`$table->integer('col')`	INTEGER カラム
`$table->boolean('col')`	BOOLEAN カラム
`$table->json('col')`	JSON カラム
`$table->timestamp('col')`	TIMESTAMP カラム
`$table->timestamps()`	created_at と updated_at
`$table->softDeletes()`	ソフトデリート用の deleted_at

ミドルウェア

カスタムミドルウェア

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

登録と使用

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

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

組み込みミドルウェア

`auth`	認証を要求
`guest`	認証済みの場合リダイレクト
`throttle:60,1`	レート制限（60 リクエスト / 分）
`verified`	メール確認を要求
`signed`	署名付き URL を検証

認証

認証ヘルパー

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

スターターキット

`Laravel Breeze`	最小限の認証スキャフォールド（Blade または Inertia）
`Laravel Jetstream`	フル機能（チーム、2FA、API トークン）
`Sanctum`	SPA / モバイル向け API トークン認証
`Passport`	完全な OAuth2 サーバー実装

ルートの保護

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

バリデーション

コントローラーバリデーション

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

フォームリクエスト

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

よく使うルール

`required`	フィールドが存在し空でないこと
`string | integer | boolean`	型バリデーション
`min:N | max:N`	最小 / 最大の長さまたは値
`email`	有効なメール形式
`unique:table,column`	DB テーブルで一意であること
`exists:table,column`	DB テーブルに存在すること
`in:a,b,c`	列挙した値のいずれかであること
`confirmed`	_confirmation フィールドとの一致を要求
`date | after:date`	日付バリデーション

よく使うパターン

API レスポンス

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

環境と設定

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

便利なヘルパー

`route('name', $params)`	名前付きルートの URL を生成
`redirect()->route('name')`	名前付きルートにリダイレクト
`back()->withErrors()`	バリデーションエラーと共に戻る
`abort(404)`	HTTP 例外をスロー
`collect($array)`	配列からコレクションを作成
`now()`	現在の Carbon 日時
`cache()->remember()`	TTL 付きで値をキャッシュ