

Référence rapide Laravel

Artisan, routage, Eloquent, Blade, middleware, auth

Artisan

Commandes courantes

<code>php artisan serve</code>	Démarrer le serveur de développement
<code>php artisan make:model Name -m</code>	Créer un modèle avec une migration
<code>php artisan make:controller NameController</code>	Créer une classe contrôleur
<code>php artisan make:middleware Name</code>	Créer une classe middleware
<code>php artisan migrate</code>	Exécuter les migrations en attente
<code>php artisan migrate:rollback</code>	Annuler le dernier lot de migrations
<code>php artisan db:seed</code>	Exécuter les seeders de base de données
<code>php artisan tinker</code>	REPL interactif pour l'application
<code>php artisan route:list</code>	Lister toutes les routes enregistrées
<code>php artisan cache:clear</code>	Vider le cache de l'application
<code>php artisan config:clear</code>	Vider la configuration en cache
<code>php artisan queue:work</code>	Démarrer le traitement des tâches en file

Routage

Routes de base

```
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']);
```

Paramètres et groupes de routes

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

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

Fonctionnalités de routage

<code>->name('route.name')</code>	Route nommée pour la génération d'URL
<code>->where('id', '[0-9]+')</code>	Contrainte regex sur un paramètre
<code>Route::resource()</code>	Routes de ressource RESTful (7 routes)
<code>Route::apiResource()</code>	Ressource API (sans vues create/edit)
<code>Route::fallback()</code>	Capture tout pour les routes non trouvées

Contrôleurs

Contrôleur de ressource

```
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');
    }
}
```

Méthodes de ressource

<code>index()</code>	GET /resource -- lister tout
<code>create()</code>	GET /resource/create -- afficher le formulaire
<code>store()</code>	POST /resource -- enregistrer
<code>show(\$id)</code>	GET /resource/{id} -- afficher un
<code>edit(\$id)</code>	GET /resource/{id}/edit -- formulaire d'édition
<code>update(\$id)</code>	PUT /resource/{id} -- mettre à jour
<code>destroy(\$id)</code>	DELETE /resource/{id} -- supprimer

Templates Blade

Mise en page et sections

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

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

Directives

<code>{{ \$var }}</code>	Afficher avec échappement HTML
<code>{!! \$html !!}</code>	Afficher brut (non échappé)
<code>@if / @elseif / @else</code>	Blocs conditionnels
<code>@foreach (\$items as \$item)</code>	Itérer sur une collection
<code>@forelse / @empty</code>	Boucle avec repli si vide
<code>@include('partial')</code>	Inclure une autre vue Blade
<code>@component / @slot</code>	Composants Blade réutilisables
<code>@csrf</code>	Champ caché du token CSRF
<code>@auth / @guest</code>	Vérifier le statut d'authentification
<code>@error('field')</code>	Afficher une erreur de validation

Eloquent ORM

Bases du modèle

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

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

Requêtes

```
Post::all(); // tous les enregistrements
Post::find(1); // par clé primaire
Post::where('status', 'published')->get();
Post::where('views', '>', 100)->orderBy('created_at', 'desc')->first();
```

Opérations CRUD

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

Relations

<code>hasOne</code>	Un-à-un (User->Phone)
<code>hasMany</code>	Un-à-plusieurs (Post->Comments)
<code>belongsTo</code>	Inverse de hasOne/hasMany
<code>belongsToMany</code>	Plusieurs-à-plusieurs avec table pivot
<code>hasManyThrough</code>	Has-many via un modèle intermédiaire

Migrations

Créer des tables

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

Types de colonnes

<code>\$table->id()</code>	Clé primaire BIGINT auto-incrémentée
<code>\$table->string('col', 100)</code>	VARCHAR avec longueur optionnelle
<code>\$table->text('col')</code>	Colonne TEXT
<code>\$table->integer('col')</code>	Colonne INTEGER
<code>\$table->boolean('col')</code>	Colonne BOOLEAN
<code>\$table->json('col')</code>	Colonne JSON
<code>\$table->timestamp('col')</code>	Colonne TIMESTAMP
<code>\$table->timestamps()</code>	created_at et updated_at
<code>\$table->softDeletes()</code>	deleted_at pour les suppressions douces

Middleware

Middleware personnalisé

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

Enregistrement et utilisation

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

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

Middleware intégrés

<code>auth</code>	Exiger l'authentification
<code>guest</code>	Rediriger si authentifié
<code>throttle:60,1</code>	Limitation de débit (60 req/min)
<code>verified</code>	Exiger la vérification email
<code>signed</code>	Valider l'URL signée

Référence rapide Laravel

Authentification

Helpers Auth

```
Auth::check(); // l'utilisateur est-il connecté ?
Auth::user(); // modèle User courant
Auth::id(); // ID de l'utilisateur courant
Auth::attempt(['email' => $e, 'password' => $p]);
Auth::logout();
```

Kits de démarrage

Laravel Breeze	Scaffold d'auth minimal (Blade ou Inertia)
Laravel Jetstream	Complet (équipes, 2FA, tokens API)
Sanctum	Authentification par token SPA / mobile
Passport	Implémentation complète de serveur OAuth2

Protéger les routes

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

Validation

Validation dans le contrôleur

```
$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',
        ];
    }
}
```

Règles courantes

required	Le champ doit être présent et non vide
string integer boolean	Validation de type
min:N max:N	Longueur ou valeur min/max
email	Format email valide
unique:table,column	Doit être unique dans la table DB
exists:table,column	Doit exister dans la table DB
in:a,b,c	Doit être l'une des valeurs listées
confirmed	Nécessite un champ _confirmation correspondant
date after:date	Validation de date

Modèles courants

Réponse API

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

Environnement et configuration

```
env('APP_KEY'); // lire une valeur .env
config('app.name'); // lire une valeur de config
config(['app.debug' => true]); // définir à l'exécution
```

Helpers utiles

route('name', \$params)	Générer l'URL pour une route nommée
redirect()->route('name')	Rediriger vers une route nommée
back()->withErrors()	Rediriger en arrière avec les erreurs de validation
abort(404)	Lancer une exception HTTP
collect(\$array)	Créer une Collection depuis un tableau
now()	Datetime Carbon actuelle
cache()->remember()	Mettre en cache une valeur avec TTL