LARAVEL 快速参考

Artisan、路由、Eloquent、Blade、中间件、认证

Artisan

常用命令

`php artisan serve`	启动开发服务器
`php artisan make:model Name -m`	创建模型并生成迁移
`php artisan make:controller NameController`	创建控制器类
`php artisan make:middleware Name`	创建中间件类
`php artisan migrate`	运行待执行的迁移
`php artisan migrate:rollback`	回滚最后一批迁移
`php artisan db:seed`	运行数据库填充
`php artisan tinker`	应用交互式 REPL
`php artisan route:list`	列出所有已注册路由
`php artisan cache:clear`	清除应用缓存
`php artisan config:clear`	清除缓存的配置
`php artisan queue:work`	开始处理队列任务

路由

基本路由

```
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']);
```

路由参数与分组

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

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

路由特性

`->name('route.name')`	命名路由，用于 URL 生成
`->where('id', '[0-9]+')`	参数的正则约束
`Route::resource()`	RESTful 资源路由（7 个路由）
`Route::apiResource()`	API 资源路由（无 create/edit 视图）
`Route::fallback()`	未匹配路由的兜底处理

控制器

资源控制器

```
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');
    }
}
```

资源方法

`index()`	GET /resource -- 列出全部
`create()`	GET /resource/create -- 显示表单
`store()`	POST /resource -- 保存新记录
`show($id)`	GET /resource/{id} -- 显示一条
`edit($id)`	GET /resource/{id}/edit -- 编辑表单
`update($id)`	PUT /resource/{id} -- 更新
`destroy($id)`	DELETE /resource/{id} -- 删除

Blade 模板

布局与区块

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

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

指令

`{{ $var }}`	输出并转义 HTML
`{!! $html !!}`	输出原始内容（不转义）
`@if / @elseif / @else`	条件块
`@foreach ($items as $item)`	遍历集合
`@forelse / @empty`	带空值回退的循环
`@include('partial')`	引入另一个 Blade 视图
`@component / @slot`	可复用的 Blade 组件
`@csrf`	CSRF 令牌隐藏字段
`@auth / @guest`	检查认证状态
`@error('field')`	显示验证错误

Eloquent ORM

模型基础

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

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

查询

```
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
```

关联关系

`hasOne`	一对一（User -> Phone）
`hasMany`	一对多（Post -> Comments）
`belongsTo`	hasOne/hasMany 的反向
`belongsToMany`	多对多（含中间表）
`hasManyThrough`	通过中间模型的远层一对多

迁移

创建数据表

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

列类型

`$table->id()`	自增 BIGINT 主键
`$table->string('col', 100)`	VARCHAR，可选长度
`$table->text('col')`	TEXT 列
`$table->integer('col')`	INTEGER 列
`$table->boolean('col')`	BOOLEAN 列
`$table->json('col')`	JSON 列
`$table->timestamp('col')`	TIMESTAMP 列
`$table->timestamps()`	created_at 和 updated_at
`$table->softDeletes()`	软删除用的 deleted_at

中间件

自定义中间件

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

注册与使用

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

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

内置中间件

`auth`	要求已认证
`guest`	已认证则重定向
`throttle:60,1`	限流（每分钟 60 次）
`verified`	要求邮箱已验证
`signed`	验证签名 URL

认证

Auth 辅助函数

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

认证套件

`Laravel Breeze`	最小化认证脚手架（Blade 或 Inertia）
`Laravel Jetstream`	全功能（团队、2FA、API Token）
`Sanctum`	SPA / 移动端 API Token 认证
`Passport`	完整 OAuth2 服务器实现

保护路由

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

验证

控制器验证

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

表单请求

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

常用规则

`required`	字段必须存在且不为空
`string | integer | boolean`	类型验证
`min:N | max:N`	最小/最大长度或值
`email`	有效的邮箱格式
`unique:table,column`	数据库表中必须唯一
`exists:table,column`	数据库表中必须存在
`in:a,b,c`	必须是列举值之一
`confirmed`	需要匹配的 _confirmation 字段
`date | after:date`	日期验证

常见模式

API 响应

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

环境与配置

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

常用辅助函数

`route('name', $params)`	生成命名路由的 URL
`redirect()->route('name')`	重定向到命名路由
`back()->withErrors()`	携带验证错误重定向回上一页
`abort(404)`	抛出 HTTP 异常
`collect($array)`	从数组创建 Collection
`now()`	当前 Carbon 日期时间
`cache()->remember()`	带 TTL 的缓存值