Laravel Riferimento Rapido

Artisan, routing, Eloquent, Blade, middleware, auth

Artisan

Comandi Comuni

`php artisan serve`	Avvia il server di sviluppo
`php artisan make:model Name -m`	Crea un model con migration
`php artisan make:controller NameController`	Crea una classe controller
`php artisan make:middleware Name`	Crea una classe middleware
`php artisan migrate`	Esegui le migration in sospeso
`php artisan migrate:rollback`	Annulla l'ultimo batch di migration
`php artisan db:seed`	Esegui i database seeder
`php artisan tinker`	REPL interattivo per l'app
`php artisan route:list`	Elenca tutte le route registrate
`php artisan cache:clear`	Pulisce la cache dell'applicazione
`php artisan config:clear`	Pulisce la config in cache
`php artisan queue:work`	Avvia l'elaborazione dei job in coda

Routing

Route di 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']);
```

Parametri e Gruppi di Route

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

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

Funzionalità delle Route

`->name('route.name')`	Route con nome per la generazione di URL
`->where('id', '[0-9]+')`	Vincolo regex sul parametro
`Route::resource()`	Route RESTful resource (7 route)
`Route::apiResource()`	Resource API (senza viste create/edit)
`Route::fallback()`	Catch-all per route non corrispondenti

Controller

Controller Resource

```
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');
    }
}
```

Metodi Resource

`index()`	GET /resource -- elenca tutti
`create()`	GET /resource/create -- mostra form
`store()`	POST /resource -- salva nuovo
`show($id)`	GET /resource/{id} -- mostra uno
`edit($id)`	GET /resource/{id}/edit -- form modifica
`update($id)`	PUT /resource/{id} -- aggiorna
`destroy($id)`	DELETE /resource/{id} -- elimina

Template Blade

Layout e Sezioni

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

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

Direttive

`{{ $var }}`	Stampa con escape HTML
`{!! $html !!}`	Stampa raw (senza escape)
`@if / @elseif / @else`	Blocchi condizionali
`@foreach ($items as $item)`	Itera su una collezione
`@forelse / @empty`	Ciclo con fallback per lista vuota
`@include('partial')`	Include un'altra vista Blade
`@component / @slot`	Componenti Blade riutilizzabili
`@csrf`	Campo nascosto con token CSRF
`@auth / @guest`	Verifica lo stato di autenticazione
`@error('field')`	Mostra errore di validazione

Eloquent ORM

Basi del Model

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

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

Query

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

Operazioni CRUD

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

Relazioni

`hasOne`	Uno-a-uno (User -> Phone)
`hasMany`	Uno-a-molti (Post -> Comments)
`belongsTo`	Inverso di hasOne/hasMany
`belongsToMany`	Molti-a-molti con tabella pivot
`hasManyThrough`	Has-many tramite model intermedio

Migration

Creazione di Tabelle

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

Tipi di Colonna

`$table->id()`	Chiave primaria BIGINT auto-incrementante
`$table->string('col', 100)`	VARCHAR con lunghezza opzionale
`$table->text('col')`	Colonna TEXT
`$table->integer('col')`	Colonna INTEGER
`$table->boolean('col')`	Colonna BOOLEAN
`$table->json('col')`	Colonna JSON
`$table->timestamp('col')`	Colonna TIMESTAMP
`$table->timestamps()`	created_at e updated_at
`$table->softDeletes()`	deleted_at per soft delete

Middleware

Middleware Personalizzato

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

Registrazione e Utilizzo

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

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

Middleware Built-in

`auth`	Richiede autenticazione
`guest`	Reindirizza se autenticato
`throttle:60,1`	Limite di frequenza (60 req/min)
`verified`	Richiede verifica email
`signed`	Valida URL firmato

Laravel Riferimento Rapido

Autenticazione

Helper Auth

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

Starter Kit

`Laravel Breeze`	Scaffolding auth minimale (Blade o Inertia)
`Laravel Jetstream`	Completo (team, 2FA, API token)
`Sanctum`	Autenticazione tramite token per SPA/mobile
`Passport`	Implementazione completa del server OAuth2

Protezione delle Route

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

Validazione

Validazione nel Controller

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

Regole Comuni

`required`	Campo obbligatorio e non vuoto
`string | integer | boolean`	Validazione del tipo
`min:N | max:N`	Lunghezza o valore minimo/massimo
`email`	Formato email valido
`unique:table,column`	Deve essere unico nella tabella DB
`exists:table,column`	Deve esistere nella tabella DB
`in:a,b,c`	Deve essere uno dei valori elencati
`confirmed`	Richiede il campo _confirmation corrispondente
`date | after:date`	Validazione della data

Pattern Comuni

Risposta API

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

Ambiente e Configurazione

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

Helper Utili

`route('name', $params)`	Genera URL per route con nome
`redirect()->route('name')`	Reindirizza a route con nome
`back()->withErrors()`	Torna indietro con errori di validazione
`abort(404)`	Lancia un'eccezione HTTP
`collect($array)`	Crea una Collection da un array
`now()`	Datetime Carbon corrente
`cache()->remember()`	Mette in cache un valore con TTL