Introduction

Properly organizing and naming routes in a Laravel application is crucial for maintaining a clean and manageable codebase. This guide covers best practices for naming conventions, grouping related routes, and how to leverage route names for easier referencing and testing in a web and API context.

Benefits of Proper Route Naming and Grouping

Clarity and Readability: Well-named routes provide a clear understanding of the functionality they represent.

Ease of Maintenance: Grouping related routes together makes it easier to manage and update them.

Route Resolution: Using named routes allows for more flexible and readable code when generating URLs or redirects.

Testing: Named routes simplify testing by providing a consistent way to refer to routes.

Naming Conventions

Adopting a consistent naming convention for routes improves readability and maintainability. Here are some guidelines:

Resource Routes: Use the resource method to define CRUD routes for a resource.

Route::resource('policies', PolicyController::class);

Plural Naming: Use plural names for route URIs to reflect that they operate on collections, and singular parameter names for individual items.

Route::get('policies', [PolicyController::class, 'index']); // Retrieves a collection of policies

Route::get('policies/{policy}', [PolicyController::class, 'show']); // Retrieves a single policy

Named Routes: Assign meaningful names to routes for easier referencing.

Route::get('policies', [PolicyController::class, 'index'])->name('policies.index');

Route::get('policies/{policy}', [PolicyController::class, 'show'])->name('policy.show');

Hyphenated Names: Avoid using underscores in two-word route names; use hyphens instead.

Route::get('policy-documents/{policy}', [PolicyController::class, 'getDocuments'])->name('policy.documents');

Edge Cases

Compound Words: Use hyphens to separate words in complex or compound names.

Example: Route::get('policy-details/{policy}', [PolicyController::class, 'details'])->name('policy.details');

Abbreviations: Avoid using abbreviations in route names to maintain clarity.

Example: Route::get('claim-adjustments/{claim}', [ClaimController::class, 'adjustments'])->name('claim.adjustments');

Nested Resources: Use nested routes for nested resources to reflect the relationship in the URI.

Example:

Route::prefix('policies/{policy}')->group(function () {

Route::get('claims', [ClaimController::class, 'index'])->name('policy.claims.index');

Route::post('claims', [ClaimController::class, 'store'])->name('policy.claims.store');

Route::get('claims/{claim}', [ClaimController::class, 'show'])->name('policy.claims.show');

});

Action-Specific Routes: Include the action in the route name for clarity.

Example: Route::post('claims/{claim}/approve', [ClaimController::class, 'approve'])->name('claim.approve');

Good and Bad Practices

Good Practices

Use Descriptive Names: Ensure route names clearly describe the action and resource.

Example:

Route::get('policies', [PolicyController::class, 'index'])->name('policies.index');

Route::get('policies/{policy}', [PolicyController::class, 'show'])->name('policy.show');

Group Related Routes: Organize routes by grouping them based on functionality or middleware requirements.

Example of Grouping Web Routes:

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\Admin\PolicyController;

Route::prefix('admin')->middleware('auth')->group(function () {

Route::get('policies', [PolicyController::class, 'index'])->name('admin.policies.index');

Route::get('policies/{policy}', [PolicyController::class, 'show'])->name('admin.policy.show');

Route::post('policies', [PolicyController::class, 'store'])->name('admin.policy.store');

Route::put('policies/{policy}', [PolicyController::class, 'update'])->name('admin.policy.update');

Route::patch('policies/{policy}', [PolicyController::class, 'partialUpdate'])->name('admin.policy.partial\_update');

Route::delete('policies/{policy}', [PolicyController::class, 'destroy'])->name('admin.policy.destroy');

});

Use Named Routes: Leverage named routes for better readability and maintainability in your application and tests.

Example: Route::get('policies', [PolicyController::class, 'index'])->name('policies.index');

Use Class References: Always use [Controller::class, 'method'] format for routes to allow easy refactoring in IDEs.

Example: Route::get('policies', [PolicyController::class, 'index']);

Use Plural Naming: Always use plural naming conventions for collections and singular parameter names for individual items.

Example: Route::get('policies/{policy}', [PolicyController::class, 'show']);

Use Hyphens in Multi-word Names: Avoid using underscores in two-word route names; use hyphens instead.

Example: Route::get('policy-details/{policy}', [PolicyController::class, 'details'])->name('policy.details');

Use Appropriate HTTP Methods:

GET policies - Fetch an array of policies.

Route::get('policies', [PolicyController::class, 'index'])->name('policies.index');

POST policies - Create a new policy.

Route::post('policies', [PolicyController::class, 'store'])->name('policies.store');

PUT policies/{policy} - Update a policy.

Route::put('policies/{policy}', [PolicyController::class, 'update'])->name('policies.update');

PATCH policies/{policy} - Partially update a policy.

Route::patch('policies/{policy}', [PolicyController::class, 'partialUpdate'])->name('policies.partial\_update');

DELETE policies/{policy} - Delete a policy.

Route::delete('policies/{policy}', [PolicyController::class, 'destroy'])->name('policies.destroy');

Bad Practices

Overly Generic Names: Avoid using generic names that do not convey the purpose of the route.

Bad Example: Route::get('view', [PolicyController::class, 'index']);

Hardcoding URLs: Do not hardcode URLs in your views or controllers. Use named routes instead.

Bad Example: return redirect('admin/policies/' . $policy->id);

Correct Example:

// Generating a URL

$url = route('admin.policies.index');

// Redirecting to a named route

return redirect()->route('admin.policy.show', ['policy' => $policy->id]);

Disorganized Routes: Avoid having all routes in a single file without grouping, as it makes maintenance difficult.

Bad Example:

Route::get('policies', [PolicyController::class, 'index']);

Route::get('policies/{policy}', [PolicyController::class, 'show']);

Route::post('policies', [PolicyController::class, 'store']);

Route::put('policies/{policy}', [PolicyController::class, 'update']);

Route::patch('policies/{policy}', [PolicyController::class, 'partialUpdate']);

Route::delete('policies/{policy}', [PolicyController::class, 'destroy']);