

Laravel - Routing

Topics : [Laravel](#)

Written on [December 18, 2023](#)

Routing in Laravel is a fundamental concept that defines how the application responds to HTTP requests. Laravel uses a simple and expressive routing syntax, allowing you to define routes for various URI patterns and link them to controller actions or closures. Here's an overview of Laravel routing:--

Basic Routing:

1. Defining Routes:

- Routes are typically defined in the `routes/web.php` file for web routes and `routes/api.php` for API routes.
- Define a route using the Route facade:

```
Route::get('/', function () { return 'Welcome to the homepage!'; });
```

This route responds to HTTP GET requests to the root URL (/) with a closure that returns a simple message.

2. HTTP Verbs:

- Laravel provides methods for common HTTP verbs (GET, POST, PUT, PATCH, DELETE). For example:

```
Route::post('/submit', 'FormController@submit');
```

This route responds to HTTP POST requests to the `/submit` URL and directs them to the `submit` method of the `FormController`.

Route Parameters:

1. Required Parameters:

- You can define route parameters by enclosing them in curly braces `{}`:

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

This route responds to URLs like `/user/123`.

2. Optional Parameters:

- You can make parameters optional by providing a default value:

```
Route::get('/user/{name?}', function ($name = 'Guest') { return 'Hello, ' . $name; });
```

This route responds to URLs like `/user` or `/user/John`.

Named Routes:

1. Naming Routes:

- You can name routes to simplify URL generation and redirects:

```
Route::get('/dashboard', 'DashboardController@index')->name('dashboard');
```

2. Generating URLs:

- Use the `route` function to generate URLs for named routes:

```
$url = route('dashboard');
```

Route Groups:

1. Grouping Routes:

- You can group routes to apply common attributes, such as middleware or a common namespace:

```
Route::middleware(['auth'])->group(function () { // Routes that require authentication });
```

Route Middleware:

1. Applying Middleware:

- Middleware can be applied to routes to perform actions before or after the request enters the controller:

```
Route::get('/admin', function () { // Your logic here })->middleware('auth');
```

2. Multiple Middleware:

- You can apply multiple middleware to a route:

```
Route::get('/admin', function () { // Your logic here }->middleware(['auth', 'admin']));
```

Route Caching:

1. Caching Routes:

- In production, you can cache routes for better performance:

```
php artisan route:cache
```

- To clear the route cache:

```
php artisan route:clear
```

Route Model Binding:

1. Implicit Binding:

- Laravel supports automatic model binding in routes:

```
Route::get('/user/{user}', function (App\Models\User $user) { return $user; });
```

The User model instance will be injected based on the route parameter.

2. Custom Binding:

- You can define custom model bindings in the RouteServiceProvider.

Route Resources:

1. Resourceful Routes:

- Use the resource method to define resourceful routes:

```
Route::resource('photos', 'PhotoController');
```

This creates routes for common CRUD operations on photos.

2. Naming Resource Routes:

- You can name resource routes for URL generation:

```
Route::resource('photos', 'PhotoController')->names('admin.photos');
```

This allows generating URLs like `route('admin.photos.index')`.

ARYATECHNO