Laravel Hidden Features You Need to Know

Laravel is a powerful PHP framework that has been around for over a decade. It has gained a massive following due to its elegant syntax, robust set of features, and ease of use. Whether you’re a seasoned Laravel developer or just starting out, there are a few hidden features that you might not be aware of. In this blog post, we’ll highlight some of the most useful Laravel hidden features that you need to know in 2023.

**1. Route caching:**

Route caching is a feature that was introduced in Laravel 5.5. It allows you to cache your application’s routes, which can significantly speed up your application’s boot time. To use route caching, simply run the following command in your terminal:

php artisan route:cache

**2. Artisan command scheduler:**

The Artisan command scheduler allows you to schedule repetitive tasks in your application, such as sending emails, cleaning up old data, or updating your database. To use the Artisan command scheduler, you’ll need to add a task to the app/Console/Kernel.php file:

protected function schedule(Schedule $schedule)

{

$schedule->command('emails:send')

->dailyAt('9:00');

}

**3. Automatic Namespace Model Detection:**

Laravel has a feature called automatic namespace model detection, which allows you to omit the namespace of your models when you are using them. For example, instead of writing:

$user = App\User::find(1);

you can simply write:

$user = User::find(1);

**4. Model factories:**

Model factories are a great way to generate fake data for testing your application. You can define a factory for a model by creating a file in the database/factories directory. For example, here's a factory for a User model:

$factory->define(User::class, function (Faker $faker) {

return [

'name' => $faker->name,

'email' => $faker->unique()->safeEmail,

'password' => bcrypt('secret'),

];

});

**5. Route Model Binding:**

Route model binding is a feature that allows you to automatically retrieve an instance of a model based on the value of a URL parameter. For example, instead of writing:

Route::get('users/{id}', function ($id) {

$user = App\User::find($id);

return view('users.show', compact('user'));

});

you can simply write:

Route::get('users/{user}', function (App\User $user) {

return view('users.show', compact('user'));

});

**6. Task Scheduling with Task Closures:**

Task scheduling in Laravel can be done using task closures, which provide a more elegant way of scheduling tasks as compared to scheduling Artisan commands. Task closures can be defined in the app/Console/Kernel.php file, and are executed on a schedule defined in the same file. Here's an example of a task closure:

$schedule->call(function () {

// Your task logic

})->daily();

**7. HTTP middleware:**

HTTP middleware are classes that can be used to filter HTTP requests before they reach your application. For example, you can use a middleware to check if a user is authenticated before allowing them to access a particular route or resource. You can define your own middleware by creating a class in the app/Http/Middleware directory. Here's an example of a simple middleware:

namespace App\Http\Middleware;

use Closure;

class CheckAge

{

public function handle($request, Closure $next)

{

if ($request->age <= 200) {

return $next($request);

}

return redirect('home');

}

}

**8. Eloquent Subqueries:**

Eloquent subqueries are a powerful feature that allows you to execute subqueries within an Eloquent query. This can be useful when you need to retrieve data based on the results of another query. Here’s an example of an Eloquent subquery:

$users = App\User::select('name')

->whereIn('id', function ($query) {

$query->select('user\_id')

->from('orders')

->whereRaw('orders.created\_at >= ?', [Carbon::yesterday()]);

})

->get();

**9. Route Prefixing:**

Route prefixing allows you to group routes under a common URL prefix. This can be useful when you need to organize your routes into different sections or categories. You can define route prefixes in the routes/web.php or routes/api.php file. Here's an example of how to use route prefixing:

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

Route::get('users', function () {

// Matches The "/admin/users" URL

});

});

**10. Query Builder Macros:**

Query builder macros are custom methods that you can add to the Laravel query builder. This can be useful when you need to reuse a particular query in multiple places throughout your application. You can define query builder macros in the app/Providers/AppServiceProvider.php file. Here's an example of a query builder macro:

use Illuminate\Support\Facades\DB;

DB::macro('getTotal', function ($table) {

return $this->table($table)->sum('total');

});

// Use the macro

$total = DB::getTotal('orders');