Laravel Tips

Awesome Laravel tips and tricks for all artisans. PR and ideas are welcome!
An idea by PovilasKorop and MarceauKa.

Hey, like these tips? Also, check out my premium Laravel courses.

Or if you want the Chinese version: 中文版本

Update 9 January 2022: Currently there are **254 tips** divided into 14 sections.

Table of Contents

DB Models and Eloquent

Reuse or clone query()

Typically, we need to query multiple time from a filtered query. So, most of the time we use `query()` method,

let's write a query for getting today created active and inactive products

```
$query = Product::query();


$today = request()->q_date ?? today();
if($today){
    $query->where('created_at', $today);
}

// lets get active and inactive products
$active_products = $query->where('status', 1)->get(); // this line modified the $query object variable
$inactive_products = $query->where('status', 0)->get(); // so here we will not find any inactive products
```

But, after getting `$active products` the `$query` will be modified. So, `$inactive_products` will not find any inactive products from `$query` and that will return blank collection every time. Cause, that will try to find inactive products from `$active_products` (`$query` will return active products only).

For solve this issue, we can query multiple time by reusing this `$query` object. So, We need to clone this `$query` before doing any `$query` modification action.

```
$active_products = (clone $query)->where('status', 1)->get(); // it will not modify the $query
$inactive_products = (clone $query)->where('status', 0)->get(); // so we will get inactive products from $query
```

Eloquent where date methods

In Eloquent, check the date with functions `whereDay()`, `whereMonth()`, `whereYear()`, `whereDate()` and `whereTime()`.

```
$products = Product::whereDate('created_at', '2018-01-31')->get();
$products = Product::whereMonth('created_at', '12')->get();
$products = Product::whereDay('created_at', '31')->get();
$products = Product::whereYear('created_at', date('Y'))->get();
$products = Product::whereTime('created_at', '=', '14:13:58')->get();
```

Increments and decrements

If you want to increment some DB column in some table, just use `increment()` function. Oh, and you can increment not only by 1, but also by some number, like 50.

```
Post::find($post_id)->increment('view_count');
User::find($user_id)->increment('points', 50);
```

No timestamp columns

If your DB table doesn't contain timestamp fields `created_at` and `updated_at`, you can specify that Eloquent model wouldn't use them, with `$timestamps = false` property.

```
class Company extends Model
{
    public $timestamps = false;
}
```

Soft-deletes: multiple restore

When using soft-deletes, you can restore multiple rows in one sentence.

```
Post::onlyTrashed()->where('author_id', 1)->restore();
```

Model all: columns

When calling Eloquent's `Model::all()`, you can specify which columns to return.

```
$users = User::all(['id', 'name', 'email']);
```

To Fail or not to Fail

In addition to `findOrFail()`, there's also Eloquent method `firstOrFail()` which will return 404 page if no records for query are found.

```
$user = User::where('email', 'povilas@laraveldaily.com')->firstOrFail();
```

Column name change

In Eloquent Query Builder, you can specify "as" to return any column with a different name, just like in plain SQL query.

```
$users = DB::table('users')->select('name', 'email as user_email')->get();
```

Map query results

After Eloquent query you can modify rows by using `map()` function in Collections.

```
$users = User::where('role_id', 1)->get()->map(function (User $user) {
    $user->some_column = some_function($user);
    return $user;
});
```

Change Default Timestamp Fields

What if you're working with non-Laravel database and your timestamp columns are named differently? Maybe, you have create_time and update_time. Luckily, you can specify them in the model, too:

```
class Role extends Model
{
    const CREATED_AT = 'create_time';
    const UPDATED_AT = 'update_time';
}
```

Quick Order by created_at

Instead of:

```
User::orderBy('created_at', 'desc')->get();
```

You can do it quicker:

```
User::latest()->get();
```

By default, `latest()` will order by `created_at`.

There is an opposite method `oldest()` which would order by `created_at` ascending:

```
User::oldest()->get();
```

Also, you can specify another column to order by. For example, if you want to use `updated_at`, you can do this:

```
$lastUpdatedUser = User::latest('updated_at')->first();
```

Automatic Column Value When Creating Records

If you want to generate some DB column value when creating record, add it to model's `boot()` method. For example, if you have a field "position" and want to assign the next available position to the new record (like `Country::max('position') + 1)`, do this:

```
class Country extends Model {
    protected static function boot()
    {
        parent::boot();

        Country::creating(function($model) {
            $model->position = Country::max('position') + 1;
        });
    }
}
```

DB Raw Query Calculations Run Faster

Use SQL raw queries like `whereRaw()` method, to make some DB-specific calculations directly in query, and not in Laravel, usually the result will be faster. Like, if you want to get users that were active 30+ days after their registration, here's the code:

```
User::where('active', 1)
    ->whereRaw('TIMESTAMPDIFF(DAY, created_at, updated_at) > ?', 30)
    ->get();
```

More than One Scope

You can combine and chain Query Scopes in Eloquent, using more than one scope in a query.

Model:

```
public function scopeActive($query) {
    return $query->where('active', 1);
}

public function scopeRegisteredWithinDays($query, $days) {
    return $query->where('created_at', '>=', now()->subDays($days));
}
```

Some Controller:

```
$users = User::registeredWithinDays(30)->active()->get();
```

No Need to Convert Carbon

If you're performing `whereDate()` and check today's records, you can use Carbon's `now()` and it will automatically be transformed to date. No need to do `->toDateString()`.

```
// Instead of
$todayUsers = User::whereDate('created_at', now()->toDateString())->get();
// No need to convert, just use now()
$todayUsers = User::whereDate('created_at', now())->get();
```

Grouping by First Letter

You can group Eloquent results by any custom condition, here's how to group by first letter of user's name:

```
$users = User::all()->groupBy(function($item) {
    return $item->name[0];
});
```

Never Update the Column

If you have DB column which you want to be set only once and never updated again, you can set that restriction on Eloquent Model, with a mutator:

```
class User extends Model
{
    public function setEmailAttribute($value)
    {
        if ($this->email) {
            return;
        }

        $this->attributes['email'] = $value;
    }
}
```

Find Many

Eloquent method `find()` may accept multiple parameters, and then it returns a Collection of all records found, not just one Model:

```
// Will return Eloquent Model
$user = User::find(1);
// Will return Eloquent Collection
$users = User::find([1,2,3]);
```

```
return Product::whereIn('id', $this->productIDs)->get();
// You can do this
return Product::find($this->productIDs)
```

Tip given by @tahiriqbalnajam

Find Many and return specific columns

Eloquent method `find()` may accept multiple parameters, and then it returns a Collection of all records found with specificied columns, not all columns of model:

```
// Will return Eloquent Model with first_name and email only
$user = User::find(1, ['first_name', 'email']);
// Will return Eloquent Collection with first_name and email only
$users = User::find([1,2,3], ['first_name', 'email']);
```

Tip given by @tahiriqbalnajam

Find by Key

You can also find multiple records with `whereKey()` method which takes care of which field is exactly your primary key (`id` is the default but you may override it in Eloquent model):

```
$users = User::whereKey([1,2,3])->get();
```

Use UUID instead of auto-increment

You don't want to use auto incrementing ID in your model?

Migration:

```
Schema::create('users', function (Blueprint $table) {
    // $table->increments('id');
    $table->uuid('id')->unique();
});
```

Model:

```
class User extends Model
{
    public $incrementing = false;
    protected $keyType = 'string';

    protected static function boot()
    {
        parent::boot();

        User::creating(function ($model) {
            $model->setId();
        });
    }

    public function setId()
    {
        $this->attributes['id'] = Str::uuid();
    }
}
```

Sub-selects in Laravel Way

From Laravel 6, you can use addSelect() in Eloquent statement, and do some calculation to that added column.

```
return Destination::addSelect(['last_flight' => Flight::select('name')
    ->whereColumn('destination_id', 'destinations.id')
    ->orderBy('arrived_at', 'desc')
    ->limit(1)
])->get();
```

Hide Some Columns

When doing Eloquent query, if you want to hide specific field from being returned, one of the quickest ways is to add `->makeHidden()` on Collection result.

```
$users = User::all()->makeHidden(['email_verified_at', 'deleted_at']);
```

Exact DB Error

If you want to catch Eloquent Query exceptions, use specific `QueryException` instead default Exception class, and you will be able to get the exact SQL code of the error.

```
try {
    // Some Eloquent/SQL statement
} catch (\Illuminate\Database\QueryException $e) {
    if ($e->getCode() === '23000') { // integrity constraint violation
        return back()->withError('Invalid data');
    }
}
```

Soft-Deletes with Query Builder

Don't forget that soft-deletes will exclude entries when you use Eloquent, but won't work if you use Query Builder.

```
// Will exclude soft-deleted entries
$users = User::all();

// Will NOT exclude soft-deleted entries
$users = User::withTrashed()->get();

// Will NOT exclude soft-deleted entries
$users = DB::table('users')->get();
```

Good Old SQL Query

If you need to execute a simple SQL query, without getting any results - like changing something in DB schema, you can just do `DB::statement()`.

```
DB::statement('DROP TABLE users');
DB::statement('ALTER TABLE projects AUTO_INCREMENT=123');
```

Use DB Transactions

If you have two DB operations performed, and second may get an error, then you should rollback the first one, right?

For that, I suggest to use DB Transactions, it's really easy in Laravel:

```
DB::transaction(function () {
    DB::table('users')->update(['votes' => 1]);

    DB::table('posts')->delete();
});
```

Update or Create

If you need to check if the record exists, and then update it, or create a new record otherwise, you can do it in one sentence - use Eloquent method `updateOrCreate()`:

```
// Instead of this
$flight = Flight::where('departure', 'Oakland')
    ->where('destination', 'San Diego')
    ->first();
if ($flight) {
    $flight->update(['price' => 99, 'discounted' => 1]);
} else {
    $flight = Flight::create([
        'departure' => 'Oakland',
        'destination' => 'San Diego',
        'price' => 99,
        'discounted' => 1
    ]);
}
// Do it in ONE sentence
$flight = Flight::updateOrCreate(
    ['departure' => 'Oakland', 'destination' => 'San Diego'],
    ['price' => 99, 'discounted' => 1]
);
```

Forget Cache on Save

Tip given by @pratiksh404

If you have cache key like `posts` that gives collection, and you want to forget that cache key on new store or update, you can call static `saved` function on your model:

```
class Post extends Model
{
    // Forget cache key on storing or updating
    public static function boot()
    {
        parent::boot();
        static::saved(function () {
           Cache::forget('posts');
        });
    }
}
```

Change Format Of Created_at and Updated_at

Tip given by @syofyanzuhad

To change the format of `created_at` you can add a method in your model like this:

```
public function getCreatedAtFormattedAttribute()
{
   return $this->created_at->format('H:i d, M Y');
}
```

So you can use it `$entry->created_at_formatted` when it's needed. It will return the `created_at` attribute like this: `04:19 23, Aug 2020`.

And also for changing format of `updated_at` attribute, you can add this method :

```
public function getUpdatedAtFormattedAttribute()
{
   return $this->updated_at->format('H:i d, M Y');
}
```

So you can use it `$entry->updated_at_formatted` when it's needed. It will return the `updated_at` attribute like this: `04:19 23, Aug 2020`.

Storing Array Type into JSON

Tip given by @pratiksh404

If you have input field which takes an array and you have to store it as a JSON, you can use `$casts` property in your model. Here `images` is a JSON attribute.

```
protected $casts = [
    'images' => 'array',
];
```

So you can store it as a JSON, but when retrieved from DB, it can be used as an array.

Make a Copy of the Model

If you have two very similar Models (like shipping address and billing address) and you need to make a copy of one to another, you can use `replicate()` method and change some properties after that.

Example from the official docs:

```
$shipping = Address::create([
    'type' => 'shipping',
    'line_1' => '123 Example Street',
    'city' => 'Victorville',
    'state' => 'CA',
    'postcode' => '90001',
]);

$billing = $shipping->replicate()->fill([
    'type' => 'billing'
]);

$billing->save();
```

Reduce Memory

Sometimes we need to load a huge amount of data into memory. For example:

```
$orders = Order::all();
```

But this can be slow if we have really huge data, because Laravel prepares objects of the Model class. In such cases, Laravel has a handy function `toBase()`

```
$orders = Order::toBase()->get();
//$orders will contain `Illuminate\Support\Collection` with objects `StdClass`.
```

By calling this method, it will fetch the data from the database, but it will not prepare the Model class. Keep in mind it is often a good idea to pass an array of fields to the get method, preventing all fields to be fetched from the database.

Force query without $fillable/$guarded

If you create a Laravel boilerplate as a "starter" for other devs, and you're not in control of what THEY would later fill in Model's $fillable/$guarded, you may use forceFill()

```
$team->update(['name' => $request->name])
```

What if "name" is not in Team model's `$fillable`? Or what if there's no `$fillable/$guarded` at all?

```
$team->forceFill(['name' => $request->name])
```

This will "ignore" the `$fillable` for that one query and will execute no matter what.

3-level structure of parent-children

If you have a 3-level structure of parent-children, like categories in an e-shop, and you want to show the number of products on the third level, you can use `with('yyy.yyy')` and then add `withCount()` as a condition

```
class HomeController extend Controller
{
    public function index()
    {
        $categories = Category::query()
            ->whereNull('category_id')
            ->with(['subcategories.subcategories' => function($query) {
                $query->withCount('products');
            }])->get();
    }
}
```

```
class Category extends Model
{
    public function subcategories()
    {
        return $this->hasMany(Category::class);
    }

    public function products()
    {
    return $this->hasMany(Product::class);
    }
}
```

```
<ul>
    @foreach($categories as $category)
        <li>
            {{ $category->name }}
            @if ($category->subcategories)
                <ul>
                @foreach($category->subcategories as $subcategory)
                    <li>
                        {{ $subcategory->name }}
                        @if ($subcategory->subcategories)
                            <ul>
                                @foreach ($subcategory->subcategories as $subcategory)
                                    <li>{{ $subcategory->name }} ({{ $subcategory->product_count }})</li>
                                @endforeach
                            </ul>
                        @endif
                    </li>
                @endforeach
                </ul>
            @endif
        </li>
    @endforeach
</ul>
```

Perform any action on failure

When looking for a record, you may want to perform some actions if it's not found. In addition to `->firstOrFail()` which just throws 404, you can perform any action on failure, just do `->firstOr(function() { ... })`

```
$model = Flight::where('legs', '>', 3)->firstOr(function () {
    // ...
})
```

Check if record exists or show 404

Don't use find() and then check if the record exists. Use findOrFail().

```
$product = Product::find($id);
if (!$product) {
    abort(404);
}
$product->update($productDataArray);
```

Shorter way

```
$product = Product::findOrFail($id); // shows 404 if not found
$product->update($productDataArray);
```

Abort if condition failed

`abort_if()` can be used as shorter way to check condition and throw an error page.

```
$product = Product::findOrFail($id);
if($product->user_id != auth()->user()->id){
    abort(403);
}
```

Shorter way

```
/* abort_if(CONDITION, ERROR_CODE) */
$product = Product::findOrFail($id);
abort_if ($product->user_id != auth()->user()->id, 403)
```

Perform any extra steps before deleting model

Tip given by @back2Lobby

We can use `Model::delete()` in the overridden delete method to perform additional steps.

```
// App\Models\User.php

public function delete(){

    //extra steps here whatever you want

    //now perform the normal deletion
    Model::delete();
}
```

Fill a column automatically while you persist data to the database

If you want to fill a column automatically while you persist data to the database (e.g: slug) use Model Observer instead of hard code it every time

```
use Illuminate\Support\Str;

class Article extends Model
{
    ...
    protected static function boot()
    {
        parent:boot();

        static::saving(function ($model) {
            $model->slug = Str::slug($model->title);
        });
    }
}
```

Tip given by @sky_0xs

Extra information about the query

You can call the `explain()` method on queries to know extra information about the query.

```
Book::where('name', 'Ruskin Bond')->explain()->dd();
```

```
Illuminate\Support\Collection {#5344
    all: [
        {#15407
            +"id": 1,
            +"select_type": "SIMPLE",
            +"table": "books",
            +"partitions": null,
            +"type": "ALL",
            +"possible_keys": null,
            +"key": null,
            +"key_len": null,
            +"ref": null,
            +"rows": 9,
            +"filtered": 11.11111164093,
            +"Extra": "Using where",
        },
    ],
}
```

Tip given by @amit_merchant

Using the doesntExist() method in Laravel

```
// This works
if ( 0 === $model->where('status', 'pending')->count() ) {
}

// But since I don't care about the count, just that there isn't one
// Laravel's exists() method is cleaner.
if ( ! $model->where('status', 'pending')->exists() ) {
}

// But I find the ! in the statement above easily missed. The
// doesntExist() method makes this statement even clearer.
if ( $model->where('status', 'pending')->doesntExist() ) {
}
```

Tip given by @ShawnHooper

Trait that you want to add to a few Models to call their boot() method automatically

If you have a Trait that you want to add to a few Models to call their `boot()` method automatically, you can call Trait's method as boot[TraitName]

```
class Transaction extends  Model
{
    use MultiTenantModelTrait;
}
```

```
class Task extends  Model
{
    use MultiTenantModelTrait;
}
```

```
trait MultiTenantModelTrait
{
    // This method's name is boot[TraitName]
    // It will be auto-called as boot() of Transaction/Task
    public static function bootMultiTenantModelTrait()
    {
        static::creating(function ($model) {
            if (!$isAdmin) {
                $isAdmin->created_by_id = auth()->id();
            }
        })
    }
}
```

There are two common ways of determining if a table is empty in Laravel

There are two common ways of determining if a table is empty in Laravel. Calling `exists()` or `count()` directly on the model!
One returns a strict true/false boolean, the other returns an integer which you can use as a falsy in conditionals.

```
public function index()
{
    if (\App\Models\User::exists()) {
        // returns boolean true or false if the table has any saved rows
    }

    if (\App\Models\User::count()) {
        // returns the count of rows in the table
    }
}
```

Tip given by @aschmelyun

How to prevent “property of non-object” error

```
// BelongsTo Default Models
// Let's say you have Post belonging to Author and then Blade code:
$post->author->name;

// Of course, you can prevent it like this:
$post->author->name ?? ''
// or
@$post->author->name

// But you can do it on Eloquent relationship level:
// this relation will return an empty App\Author model if no author is attached to the post
public function author() {
    return $this->belongsTo('App\Author')->withDefault();
}
// or
public function author() {
    return $this->belongsTo('App\Author')->withDefault([
        'name' => 'Guest Author'
    ]);
}
```

Tip given by @coderahuljat

Get original attributes after mutating an Eloquent record

Get original attributes after mutating an Eloquent record you can get the original attributes by calling getOriginal()

```
$user = App\User::first();
$user->name; // John
$user->name = "Peter"; // Peter
$user->getOriginal('name'); // John
$user->getOriginal(); // Original $user record
```

Tip given by @devThaer

A simple way to seed a database

A simple way to seed a database in Laravel with a .sql dump file

```
DB::unprepared(
    file_get_contents(__DIR__ . './dump.sql')
);
```

Tip given by @w3Nicolas

The crossJoinSub method of the query constructor

Using the CROSS JOIN subquery

```
use Illuminate\Support\Facades\DB;

$totalQuery = DB::table('orders')->selectRaw('SUM(price) as total');

DB::table('orders')
    ->select('*')
    ->crossJoinSub($totalQuery, 'overall')
    ->selectRaw('(price / overall.total) * 100 AS percent_of_total')
    ->get();
```

Tip given by @PascalBaljet

Belongs to Many Pivot table naming

To determine the table name of the relationship's intermediate table, Eloquent will join the two related model names in alphabetical order.

This would mean a join between `Post` and `Tag` could be added like this:

```
class Post extends Model
{
    public $table = 'posts';

    public function tags()
    {
        return $this->belongsToMany(Tag::class);
    }
}
```

However, you are free to override this convention, and you would need to specify the join table in the second argument.

```
class Post extends Model
{
    public $table = 'posts';

    public function tags()
    {
        return $this->belongsToMany(Tag::class, 'posts_tags');
    }
}
```

If you wish to be explicit about the primary keys you can also supply these as third and fourth arguments.

```
class Post extends Model
{
    public $table = 'posts';

    public function tags()
    {
        return $this->belongsToMany(Tag::class, 'post_tag', 'post_id', 'tag_id');
    }
}
```

Tip given by @iammikek

Order by Pivot Fields

`BelongsToMany::orderByPivot()` allows you to directly sort the results of a BelongsToMany relationship query.

```
class Tag extends Model
{
    public $table = 'tags';
}

class Post extends Model
{
    public $table = 'posts';

    public function tags()
    {
        return $this->belongsToMany(Tag::class, 'post_tag', 'post_id', 'tag_id')
            ->using(PostTagPivot::class)
            ->withTimestamps()
            ->withPivot('flag');
    }
}

class PostTagPivot extends Pivot
{
    protected $table = 'post_tag';
}

// Somewhere in the Controller
public function getPostTags($id)
{
    return Post::findOrFail($id)->tags()->orderByPivot('flag', 'desc')->get();
}
```

Tip given by @PascalBaljet

Find a single record from a database

The `sole()` method will return only one record that matches the criteria. If no such entry is found, then a `NoRecordsFoundException` will be thrown. If multiple records are found, then a `MultipleRecordsFoundException` will be thrown.

```
DB::table('products')->where('ref', '#123')->sole();
```

Tip given by @PascalBaljet

Automatic records chunking

Similar to `each()` method, but easier to use. Automatically splits the result into parts (chunks).

```
return User::orderBy('name')->chunkMap(fn ($user) => [
    'id' => $user->id,
    'name' => $user->name,
]), 25);
```

Tip given by @PascalBaljet

Updating the model without dispatching events

Sometimes you need to update the model without sending any events. We can now do this with the `updateQuietly()` method, which under the hood uses the `saveQuietly()` method.

```
$flight->updateQuietly(['departed' => false]);
```

Tip given by @PascalBaljet

Periodic cleaning of models from obsolete records

To periodically clean models of obsolete records. With this trait, Laravel will do this automatically, only you need to adjust the frequency of the `model:prune` command in the Kernel class.

```
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Prunable;
class Flight extends Model
{
    use Prunable;
    /**
     * Get the prunable model query.
     *
     * @return \Illuminate\Database\Eloquent\Builder
     */
    public function prunable()
    {
        return static::where('created_at', '<=', now()->subMonth());
    }
}
```

Also, in the pruning method, you can set the actions that must be performed before deleting the model:

```
protected function pruning()
{
    // Removing additional resources,
    // associated with the model. For example, files.

    Storage::disk('s3')->delete($this->filename);
}
```

Tip given by @PascalBaljet

Immutable dates and casting to them

Laravel 8.53 introduces the `immutable_date` and `immutable_datetime` castes that convert dates to `Immutable`.

Cast to CarbonImmutable instead of a regular Carbon instance.

```
class User extends Model
{
    public $casts = [
        'date_field'     => 'immutable_date',
        'datetime_field' => 'immutable_datetime',
    ];
}
```

Tip given by @PascalBaljet

The findOrFail method also accepts a list of ids

The findOrFail method also accepts a list of ids. If any of these ids are not found, then it "fails".
Nice if you need to retrieve a specific set of models and don't want to have to check that the count you got was the count you expected

```
User::create(['id' => 1]);
User::create(['id' => 2);
User::create(['id' => 3]);

// Retrives the user...
$user = User::findOrFail(1);

// Throws a 404 because the user doesn't exist...
User::findOrFail(99);

// Retrives all 3 users...
$users = User::findOrFail([1, 2, 3]);

// Throws because it is unable to find *all* of the users
User::findOrFail([1, 2, 3, 99]);
```

Tip given by @timacdonald87

Prunable trait to automatically remove models from your database

New in Laravel 8.50: You can use the Prunable trait to automatically remove models from your database. For example, you can permanently remove soft deleted models after a few days.

```
class File extends Model
{
    use SoftDeletes;

    // Add Prunable trait
    use Prunable;

    public function prunable()
    {
        // Files matching this query will be pruned
        return static::query()->where('deleted_at', '<=', now()->subDays(14));
    }

    protected function pruning()
    {
        // Remove the file from s3 before deleting the model
        Storage::disk('s3')->delete($this->filename);
    }
}

// Add PruneCommand to your schedule (app/Console/Kernel.php)
$schedule->command(PruneCommand::class)->daily();
```

Tip by @Philo01

withAggregate method

Under the hood, the withAvg/withCount/withSum and other methods in Eloquent use the 'withAggregate' method. You can use this method to add a subselect based on a relationship

```
// Eloquent Model
class Post extends Model
{
    public function user()
    {
        return $this->belongsTo(User::class);
    }
}

// Instead of eager loading all users...
$posts = Post::with('user')->get();

// You can add a subselect to only retrieve the user's name...
$posts = Post::withAggregate('user', 'name')->get();

// This will add a 'user_name' attribute to the Post instance:
$posts->first()->user_name;
```

Tip given by @pascalbaljet

Date convention

Using the `something_at` convention instead of just a boolean in Laravel models gives you visibility into when a flag was changed – like when a product went live.

```
// Migration
Schema::table('products', function (Blueprint $table) {
    $table->datetime('live_at')->nullable();
});

// In your model
public function live()
{
    return !is_null($this->live_at);
}

// Also in your model
protected $dates = [
    'live_at'
];
```

Tip given by @alexjgarrett

Eloquent multiple upserts

The upsert() method will insert or update multiple records.

- First array: the values to insert or update
- Second: unique identifier columns used in the select statement
- Third: columns that you want to update if the record exists

```
Flight::upsert([
    ['departure' => 'Oakland', 'destination' => 'San Diego', 'price' => 99],
    ['departure' => 'Chicago', 'destination' => 'New York', 'price' => 150],
], ['departure', 'destination'], ['price']);
```

Tip given by @mmartin_joo

Retrieve the Query Builder after filtering the results

To retrieve the Query Builder after filtering the results: you can use `->toQuery()`.
The method internally use the first model of the collection and a `whereKey` comparison on the Collection models.

```
// Retrieve all logged_in users
$loggedInUsers = User::where('logged_in', true)->get();

// Filter them using a Collection method or php filtering
$nthUsers = $loggedInUsers->nth(3);

// You can't do this on the collection
$nthUsers->update(/* ... */);

// But you can retrieve the Builder using ->toQuery()
if ($nthUsers->isNotEmpty()) {
    $nthUsers->toQuery()->update(/* ... */);
}
```

Tip given by @RBilloir

Custom casts

You can create custom casts to have Laravel automatically format your Eloquent model data. Here's an example that capitalises a user's name when it is retrieved or changed.

```
class CapitalizeWordsCast implements CastsAttributes
{
    public function get($model, string $key, $value, array $attributes)
    {
        return ucwords($value);
    }

    public function set($model, string $key, $value, array $attributes)
    {
        return ucwords($value);
    }
}

class User extends Model
{
    protected $casts = [
        'name'  => CapitalizeWordsCast::class,
        'email' => 'string',
    ];
}
```

Tip given by @mattkingshott

Order based on a related model's average or count

Did you ever need to order based on a related model's average or count?
It's easy with Eloquent!

```
public function bestBooks()
{
    Book::query()
        ->withAvg('ratings as average_rating', 'rating')
        ->orderByDesc('average_rating');
}
```

Tip given by @mmartin_joo

Return transactions result

If you have a DB transaction and want to return its result, there are at least two ways, see the example

```
// 1. You can pass the parameter by reference
$invoice = NULL;
DB::transaction(function () use (&$invoice) {
    $invoice = Invoice::create(...);
    $invoice->items()->attach(...);
})

// 2. Or shorter: just return trasaction result
$invoice = DB::transaction(function () {
    $invoice = Invoice::create(...);
    $invoice->items()->attach(...);

    return $invoice;
});
```

Remove several global scopes from query

When using Eloquent Global Scopes, you not only can use MULTIPLE scopes, but also remove certain scopes when you don't need them, by providing the array to `withoutGlobalScopes()`
Link to docs

```
// Remove all of the global scopes...
User::withoutGlobalScopes()->get();

// Remove some of the global scopes...
User::withoutGlobalScopes([
    FirstScope::class, SecondScope::class
])->get();
```

Order JSON column attribute

With Eloquent you can order results by a JSON column attribute

```
// JSON column example:
// bikes.settings = {"is_retired": false}
$bikes = Bike::where('athlete_id', $this->athleteId)
        ->orderBy('name')
        ->orderByDesc('settings->is_retired')
        ->get();
```

Tip given by @brbcoding

Get single column's value from the first result

You can use `value()` method to get single column's value from the first result of a query

```
// Instead of
Integration::where('name', 'foo')->first()->active;

// You can use
Integration::where('name', 'foo')->value('active');

// or this to throw an exception if no records found
Integration::where('name', 'foo')->valueOrFail('active')';
```

Tip given by @justsanjit

Check if altered value changed key

Ever wanted to know if the changes you've made to a model have altered the value for a key? No problem, simply reach for originalIsEquivalent.

```
$user = User::first(); // ['name' => "John']

$user->name = 'John';

$user->originalIsEquivalent('name'); // true

$user->name = 'David'; // Set directly
$user->fill(['name' => 'David']); // Or set via fill

$user->originalIsEquivalent('name'); // false
```

Tip given by @mattkingshott

New way to define accessor and mutator

New way to define attribute accessors and mutators in Laravel 8.77:

```
// Before, two-method approach
public function setTitleAttribute($value)
{
    $this->attributes['title'] = strtolower($value);
}
public function getTitleAttribute($value)
{
    return strtoupper($value);
}

// New approach
protected function title(): Attribute
{
    return new Attribute(
        get: fn ($value) => strtoupper($value),
        set: fn ($value) => strtolower($value),
}
```

Tip given by @Teacoders

Models Relations

OrderBy on Eloquent relationships

You can specify orderBy() directly on your Eloquent relationships.

```
public function products()
{
    return $this->hasMany(Product::class);
}

public function productsByName()
{
    return $this->hasMany(Product::class)->orderBy('name');
}
```

Conditional relationships

If you notice that you use same relationship often with additional "where" condition, you can create a separate relationship method.

Model:

```
public function comments()
{
    return $this->hasMany(Comment::class);
}

public function approved_comments()
{
    return $this->hasMany(Comment::class)->where('approved', 1);
}
```

Raw DB Queries: havingRaw()

You can use RAW DB queries in various places, including `havingRaw()` function after `groupBy()`.

```
Product::groupBy('category_id')->havingRaw('COUNT(*) > 1')->get();
```

Eloquent has() deeper

You can use Eloquent `has()` function to query relationships even two layers deep!

```
// Author -> hasMany(Book::class);
// Book -> hasMany(Rating::class);
$authors = Author::has('books.ratings')->get();
```

Has Many. How many exactly?

In Eloquent `hasMany()` relationships, you can filter out records that have X amount of children records.

```
// Author -> hasMany(Book::class)
$authors = Author::has('books', '>', 5)->get();
```

Default model

You can assign a default model in `belongsTo` relationship, to avoid fatal errors when calling it like `{{ $post->user->name }}` if $post->user doesn't exist.

```
public function user()
{
    return $this->belongsTo('App\User')->withDefault();
}
```

Use hasMany to create Many

If you have `hasMany()` relationship, you can use `saveMany()` to save multiple "child" entries from your "parent" object, all in one sentence.

```
$post = Post::find(1);
$post->comments()->saveMany([
    new Comment(['message' => 'First comment']),
    new Comment(['message' => 'Second comment']),
]);
```

Multi level Eager Loading

In Laravel you can Eager Load multiple levels in one statement, in this example we not only load the author relation but also the country relation on the author model.

```
$users = App\Book::with('author.country')->get();
```

Eager Loading with Exact Columns

You can do Laravel Eager Loading and specify the exact columns you want to get from the relationship.

```
$users = App\Book::with('author:id,name')->get();
```

You can do that even in deeper, second level relationships:

```
$users = App\Book::with('author.country:id,name')->get();
```

Touch parent updated_at easily

If you are updating a record and want to update the `updated_at` column of parent relationship (like, you add new post comment and want `posts.updated_at` to renew), just use `$touches = ['post'];` property on child model.

```
class Comment extends Model
{
    protected $touches = ['post'];
}
```

Always Check if Relationship Exists

Never **ever** do `$model->relationship->field` without checking if relationship object still exists.

It may be deleted for whatever reason, outside your code, by someone else's queued job etc. Do `if-else`, or `{{ $model->relationship->field ?? '' }}` in Blade, or `{{ optional($model->relationship)->field }}`. With php8 you can even use the nullsafe operator `{{ $model->relationship?->field) }}`

Use withCount() to Calculate Child Relationships Records

If you have `hasMany()` relationship, and you want to calculate "children" entries, don't write a special query. For example, if you have posts and comments on your User model, write this `withCount()`:

```
public function index()
{
    $users = User::withCount(['posts', 'comments'])->get();
    return view('users', compact('users'));
}
```

And then, in your Blade file, you will access those number with `{relationship}_count` properties:

```
@foreach ($users as $user)
<tr>
    <td>{{ $user->name }}</td>
    <td class="text-center">{{ $user->posts_count }}</td>
    <td class="text-center">{{ $user->comments_count }}</td>
</tr>
@endforeach
```

You may also order by that field:

```
User::withCount('comments')->orderBy('comments_count', 'desc')->get();
```

Extra Filter Query on Relationships

If you want to load relationship data, you can specify some limitations or ordering in a closure function. For example, if you want to get Countries with only three of their biggest cities, here's the code.

```
$countries = Country::with(['cities' => function($query) {
    $query->orderBy('population', 'desc');
    $query->take(3);
}])->get();
```

Load Relationships Always, but Dynamically

You can not only specify what relationships to ALWAYS load with the model, but you can do it dynamically, in the constructor method:

```
class ProductTag extends Model
{
    protected $with = ['product'];

    public function __construct() {
        parent::__construct();
        $this->with = ['product'];

        if (auth()->check()) {
            $this->with[] = 'user';
        }
    }
}
```

Instead of belongsTo, use hasMany

For `belongsTo` relationship, instead of passing parent's ID when creating child record, use `hasMany` relationship to make a shorter sentence.

```
// if Post -> belongsTo(User), and User -> hasMany(Post)...
// Then instead of passing user_id...
Post::create([
    'user_id' => auth()->id(),
    'title' => request()->input('title'),
    'post_text' => request()->input('post_text'),
]);

// Do this
auth()->user()->posts()->create([
    'title' => request()->input('title'),
    'post_text' => request()->input('post_text'),
]);
```

Rename Pivot Table

If you want to rename "pivot" word and call your relationship something else, you just use `->as('name')` in your relationship.

Model:

```
public function podcasts() {
    return $this->belongsToMany('App\Podcast')
        ->as('subscription')
        ->withTimestamps();
}
```

Controller:

```
$podcasts = $user->podcasts();
foreach ($podcasts as $podcast) {
    // instead of $podcast->pivot->created_at ...
    echo $podcast->subscription->created_at;
}
```

Update Parent in One Line

If you have a `belongsTo()` relationship, you can update the Eloquent relationship data in the same sentence:

```
// if Project -> belongsTo(User::class)
$project->user->update(['email' => 'some@gmail.com']);
```

Laravel 7+ Foreign Keys

From Laravel 7, in migrations you don't need to write two lines for relationship field - one for the field and one for foreign key. Use method `foreignId()`.

```
// Before Laravel 7
Schema::table('posts', function (Blueprint $table)) {
    $table->unsignedBigInteger('user_id');
    $table->foreign('user_id')->references('id')->on('users');
}

// From Laravel 7
Schema::table('posts', function (Blueprint $table)) {
    $table->foreignId('user_id')->constrained();
}

// Or, if your field is different from the table reference
Schema::table('posts', function (Blueprint $table)) {
    $table->foreignId('created_by_id')->constrained('users', 'column');
}
```

Combine Two "whereHas"

In Eloquent, you can combine `whereHas()` and `orDoesntHave()` in one sentence.

```
User::whereHas('roles', function($query) {
    $query->where('id', 1);
})
->orDoesntHave('roles')
->get();
```

Check if Relationship Method Exists

If your Eloquent relationship names are dynamic and you need to check if relationship with such name exists on the object, use PHP function `method_exists($object, $methodName)`

```
$user = User::first();
if (method_exists($user, 'roles')) {
    // Do something with $user->roles()->...
}
```

Pivot Table with Extra Relations

In many-to-many relationship, your pivot table may contain extra fields, and even extra relationships to other Model.

Then generate a separate Pivot Model:

```
php artisan make:model RoleUser --pivot
```

Next, specify it in `belongsToMany()` with `->using()` method. Then you could do magic, like in the example.

```
// in app/Models/User.php
public function roles()
{
    return $this->belongsToMany(Role::class)
        ->using(RoleUser::class)
        ->withPivot(['team_id']);
}

// app/Models/RoleUser.php: notice extends Pivot, not Model
use Illuminate\Database\Eloquent\Relations\Pivot;

class RoleUser extends Pivot
{
    public function team()
    {
        return $this->belongsTo(Team::class);
    }
}

// Then, in Controller, you can do:
$firstTeam = auth()->user()->roles()->first()->pivot->team->name;
```

Load Count on-the-fly

In addition to Eloquent's `withCount()` method to count related records, you can also load the count on-the-fly, with `loadCount()`:

```
// if your Book hasMany Reviews...
$book = App\Book::first();

$book->loadCount('reviews');
// Then you get access to $book->reviews_count;

// Or even with extra condition
$book->loadCount(['reviews' => function ($query) {
    $query->where('rating', 5);
}]);
```

Randomize Relationship Order

You can use `inRandomOrder()` to randomize Eloquent query result, but also you can use it to randomize the**relationship** entries you're loading with query.

```
// If you have a quiz and want to randomize questions...

// 1. If you want to get questions in random order:
$questions = Question::inRandomOrder()->get();

// 2. If you want to also get question options in random order:
$questions = Question::with(['answers' => function($q) {
    $q->inRandomOrder();
}])->inRandomOrder()->get();
```

Filter hasMany relationships

Just a code example from my project, showing the possibility of filtering hasMany relationships.
TagTypes -> hasMany Tags -> hasMany Examples
And you wanna query all the types, with their tags, but only those that have examples, ordering by most examples.

```
$tag_types = TagType::with(['tags' => function ($query) {
    $query->has('examples')
        ->withCount('examples')
        ->orderBy('examples_count', 'desc');
}])->get();
```

Filter by many-to-many relationship pivot column

If you have a many-to-many relationship, and you add an extra column to the pivot table, here's how you can order by it when querying the list.

```
class Tournament extends Model
{
    public function countries()
    {
        return $this->belongsToMany(Country::class)->withPivot(['position']);
    }
}
```

```
class TournamentsController extends Controller

public function whatever_method() {
    $tournaments = Tournament::with(['countries' => function($query) {
            $query->orderBy('position');
        }])->latest()->get();
}
```

A shorter way to write whereHas

Released in Laravel 8.57: a shorter way to write whereHas() with a simple condition inside.

```
// Before
User::whereHas('posts', function ($query) {
    $query->where('published_at', '>', now());
})->get();

// After
User::whereRelation('posts', 'published_at', '>', now())->get();
```

You can add conditions to your relationships

```
class User
{
    public function posts()
    {
        return $this->hasMany(Post::class);
    }

    // with a getter
    public function getPublishedPostsAttribute()
    {
        return $this->posts->filter(fn ($post) => $post->published);
    }

    // with a relationship
    public function publishedPosts()
    {
        return $this->hasMany(Post::class)->where('published', true);
    }
}
```

Tip given by @anwar_nairi

New `whereBelongsTo()` Eloquent query builder method

Laravel 8.63.0 ships with a new `whereBelongsTo()` Eloquent query builder method. Smiling face with heart-shaped eyes
This allows you to remove BelongsTo foreign key names from your queries, and use the relationship method as a single source of truth instead!

```
// From:
$query->where('author_id', $author->id)

// To:
$query->whereBelongsTo($author)

// Easily add more advanced filtering:
Post::query()
    ->whereBelongsTo($author)
    ->whereBelongsTo($cateogry)
    ->whereBelongsTo($section)
    ->get();

// Specify a custom relationship:
$query->whereBelongsTo($author, 'author')
```

Tip given by @danjharrin

The `is()` method of one-to-one relationships for comparing models

We can now make comparisons between related models without further database access.

```
// BEFORE: the foreign key is taken from the Post model
$post->author_id === $user->id;

// BEFORE: An additional request is made to get the User model from the Author relationship
$post->author->is($user);

// AFTER
$post->author()->is($user);
```

Tip given by @PascalBaljet

`whereHas()` multiple connections

```
// User Model
class User extends Model
{
    protected $connection = 'conn_1';

    public function posts()
    {
        return $this->hasMany(Post::class);
    }
}

// Post Model
class Post extends Model
{
    protected $connection = 'conn_2';

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

// wherehas()
$posts = Post::whereHas('user', function ($query) use ($request) {
      $query->from('db_name_conn_1.users')->where(...);
  })->get();
```

Tip given by @adityaricki

Update an existing pivot record

If you want to update an existing pivot record on the table, use `updateExistingPivot` instead of `syncWithPivotValues`.

```
// Migrations
Schema::create('role_user', function ($table) {
    $table->unsignedId('user_id');
    $table->unsignedId('role_id');
    $table->timestamp('assigned_at');
})

// first param for the record id
// second param for the pivot records
$user->roles()->updateExistingPivot(
    $id, ['assigned_at' => now()],
);
```

Tip given by @sky_0xs

Relation that will get the newest (or oldest) item

New in Laravel 8.42: In an Eloquent model can define a relation that will get the newest (or oldest) item of another relation.

```
public function historyItems(): HasMany
{
    return $this
        ->hasMany(ApplicationHealthCheckHistoryItem::class)
        ->orderByDesc('created_at');
}

public function latestHistoryItem(): HasOne
{
    return $this
        ->hasOne(ApplicationHealthCheckHistoryItem::class)
        ->latestOfMany();
}
```

Migrations

Unsigned Integer

For foreign key migrations instead of `integer()` use `unsignedInteger()` type or `integer()->unsigned()`, otherwise you may get SQL errors.

```
Schema::create('employees', function (Blueprint $table) {
    $table->unsignedInteger('company_id');
    $table->foreign('company_id')->references('id')->on('companies');
    // ...
});
```

You can also use `unsignedBigInteger()` if that other column is `bigInteger()` type.

```
Schema::create('employees', function (Blueprint $table) {
    $table->unsignedBigInteger('company_id');
});
```

Order of Migrations

If you want to change the order of DB migrations, just rename the file's timestamp, like from `2018_08_04_070443_create_posts_table.php` to `2018_07_04_070443_create_posts_table.php` (changed from `2018_08_04` to `2018_07_04`).

They run in alphabetical order.

Migration fields with timezones

Did you know that in migrations there's not only `timestamps()` but also `timestampsTz()`, for the timezone?

```
Schema::create('employees', function (Blueprint $table) {
    $table->increments('id');
    $table->string('name');
    $table->string('email');
    $table->timestampsTz();
});
```

Also, there are columns `dateTimeTz()`, `timeTz()`, `timestampTz()`, `softDeletesTz()`.

Database migrations column types

There are interesting column types for migrations, here are a few examples.

```
$table->geometry('positions');
$table->ipAddress('visitor');
$table->macAddress('device');
$table->point('position');
$table->uuid('id');
```

See all column types on the official documentation.

Default Timestamp

While creating migrations, you can use `timestamp()` column type with option `useCurrent()` and `useCurrentOnUpdate()`, it will set `CURRENT_TIMESTAMP` as default value.

```
$table->timestamp('created_at')->useCurrent();
$table->timestamp('updated_at')->useCurrentOnUpdate();
```

Migration Status

If you want to check what migrations are executed or not yet, no need to look at the database "migrations" table, you can launch `php artisan migrate:status` command.

Example result:

```
+------+-----------------------------------------------+-------+
| Ran? | Migration                                     | Batch |
+------+-----------------------------------------------+-------+
| Yes  | 2014_10_12_000000_create_users_table          | 1     |
| Yes  | 2014_10_12_100000_create_password_resets_table | 1     |
| No   | 2019_08_19_000000_create_failed_jobs_table    |       |
+------+-----------------------------------------------+-------+
```

Create Migration with Spaces

When typing `make:migration` command, you don't necessarily have to use underscore `_` symbol between parts, like `create_transactions_table`. You can put the name into quotes and then use spaces instead of underscores.

```
// This works
php artisan make:migration create_transactions_table

// But this works too
php artisan make:migration "create transactions table"
```

Source: Steve O on Twitter

Create Column after Another Column

Notice: Only MySQL

If you're adding a new column to the existing table, it doesn't necessarily have to become the last in the list. You can specify after which column it should be created:

```
Schema::table('users', function (Blueprint $table) {
    $table->string('phone')->after('email');
});
```

If you're adding a new column to the existing table, it doesn't necessarily have to become the last in the list. You can specify before which column it should be created:

```
Schema::table('users', function (Blueprint $table) {
    $table->string('phone')->before('created_at');
});
```

If you want your column to be the first in your table , then use the first method.

```
Schema::table('users', function (Blueprint $table) {
    $table->string('uuid')->first();
});
```

Also the `after()` method can now be used to add multiple fields.

```
Schema::table('users', function (Blueprint $table) {
    $table->after('remember_token', function ($table){
        $table->string('card_brand')->nullable();
        $table->string('card_last_four', 4)->nullable();
    });
});
```

Make migration for existing table

If you make a migration for existing table, and you want Laravel to generate the Schema::table() for you, then add "_in_xxxxx_table" or "_to_xxxxx_table" at the end, or specify "--table" parameter. `php artisan change_fields_products_table` generates empty class

```
class ChangeFieldsProductsTable extends Migration
{
    public function up()
    {
        //
    }
}
```

But add `in_xxxxx_table` `php artisan make:migration change_fields_in_products_table` and it generates class with `Schemma::table()` pre-fileed

```
class ChangeFieldsProductsTable extends Migration
{
    public function up()
    {
        Schema::table('products', function (Blueprint $table) {
            //
        })
    };
}
```

Also you can specify `--table` parameter `php artisan make:migration whatever_you_want --table=products`

```
class WhateverYouWant extends Migration
{
    public function up()
    {
        Schema::table('products', function (Blueprint $table) {
            //
        })
    };
}
```

Output SQL before running migrations

When typing `migrate --pretend` command, you get the SQL query that will be executed in the terminal. It's an interesting way to debug SQL if necessary.

```
// Artisan command
php artisan migrate --pretend
```

Anonymous Migrations

The Laravel team released Laravel 8.37 with anonymous migration support, which solves a GitHub issue with migration class name collisions. The core of the problem is that if multiple migrations have the same class name, it'll cause issues when trying to recreate the database from scratch. Here's an example from the pull request tests:

```
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    public function up(
    {
        Schema::table('people', function (Blueprint $table) {
            $table->string('first_name')->nullable();
        });
    }

    public function down()
    {
        Schema::table('people', function (Blueprint $table) {
            $table->dropColumn('first_name');
        });
    }
};
```

Tip given by @nicksdot

You can add "comment" about a column inside your migrations

You can add "comment" about a column inside your migrations and provide useful information.
If database is managed by someone other than developers, they can look at comments in Table structure before performing any operations.

```
$table->unsignedInteger('interval')
    ->index()
    ->comment('This column is used for indexing.')
```

Tip given by @nicksdot

Checking For Table / Column Existence

You may check for the existence of a table or column using the hasTable and hasColumn methods:

```
if (Schema::hasTable('users')) {
    // The "users" table exists...
}

if (Schema::hasColumn('users', 'email')) {
    // The "users" table exists and has an "email" column...
}
```

Tip given by @dipeshsukhia

Views

$loop variable in foreach

Inside of foreach loop, check if current entry is first/last by just using `$loop` variable.

```
@foreach ($users as $user)
     @if ($loop->first)
        This is the first iteration.
     @endif

     @if ($loop->last)
        This is the last iteration.
     @endif

     <p>This is user {{ $user->id }}</p>
@endforeach
```

There are also other properties like `$loop->iteration` or `$loop->count`. Learn more on the official documentation.

Does view file exist?

You can check if View file exists before actually loading it.

```
if (view()->exists('custom.page')) {
 // Load the view
}
```

You can even load an array of views and only the first existing will be actually loaded.

```
return view()->first(['custom.dashboard', 'dashboard'], $data);
```

Error code Blade pages

If you want to create a specific error page for some HTTP code, like 500 - just create a blade file with this code as filename, in `resources/views/errors/500.blade.php`, or `403.blade.php` etc, and it will automatically be loaded in case of that error code.

View without controllers

If you want route to just show a certain view, don't create a Controller method, just use `Route::view()` function.

```
// Instead of this
Route::get('about', 'TextsController@about');
// And this
class TextsController extends Controller
{
    public function about()
    {
        return view('texts.about');
    }
}
// Do this
Route::view('about', 'texts.about');
```

Blade @auth

Instead of if-statement to check logged in user, use `@auth` directive.

Typical way:

```
@if(auth()->user())
    // The user is authenticated.
@endif
```

Shorter:

```
@auth
    // The user is authenticated.
@endauth
```

The opposite is `@guest` directive:

```
@guest
    // The user is not authenticated.
@endguest
```

Two-level $loop variable in Blade

In Blade's foreach you can use $loop variable even in two-level loop to reach parent variable.

```
@foreach ($users as $user)
    @foreach ($user->posts as $post)
        @if ($loop->parent->first)
            This is first iteration of the parent loop.
        @endif
    @endforeach
@endforeach
```

Create Your Own Blade Directive

It's very easy - just add your own method in `app/Providers/AppServiceProvider.php`. For example, if you want to have this for replace `<br>` tags with new lines:

```
<textarea>@br2nl($post->post_text)</textarea>
```

Add this directive to AppServiceProvider's `boot()` method:

```
public function boot()
{
    Blade::directive('br2nl', function ($string) {
        return "<?php echo preg_replace('/\<br(\s*)?\/?\>/i', \"\n\", $string); ?>";
    });
}
```

Blade Directives: IncludeIf, IncludeWhen, IncludeFirst

If you are not sure whether your Blade partial file actually would exist, you may use these condition commands:

This will load header only if Blade file exists

```
@includeIf('partials.header')
```

This will load header only for user with role_id 1

```
@includeWhen(auth()->user()->role_id == 1, 'partials.header')
```

This will try to load adminlte.header, if missing - will load default.header

```
@includeFirst('adminlte.header', 'default.header')
```

Use Laravel Blade-X variable binding to save even more space

```
 // Using include, the old way
@include("components.post", ["title" => $post->title])

// Using Blade-X
<x-post link="{{ $post->title }}" />

// Using Blade-X variable binding
<x-post :link="$post->title" />
```

Tip given by @anwar_nairi

Blade components props

```
 // button.blade.php
@props(['rounded' => false])

<button {{ $attributes->class([
    'bg-red-100 text-red-800',
    'rounded' => $rounded
    ]) }}>
    {{ $slot }}
</button>

// view.blade.php
// Non-rounded:
<x-button>Submit</x-button>

// Rounded:
<x-button rounded>Submit</x-button>
```

Tip given by @godismyjudge95

Blade Autocomplete typehint

```
@php
    /* @var App\Models\User $user */
@endphp

<div>
    // your ide will typehint the property for you
    {{$user->email}}
</div>
```

Tip given by @freekmurze

Component Syntax Tip

Did you know that if you pass colon (:) before the component parameter, you can directly pass variables without print statement `{{ }}`?

```
<x-navbar title="{{ $title }}"/>

// you can do instead

<x-navbar :title="$title"/>
```

Tip given by @sky_0xs

Automatically highlight nav links

Automatically highlight nav links when exact URL matches, or pass a path or route name pattern.
A Blade component with request and CSS classes helpers makes it ridiculously simple to show active/inactive state.

```
class NavLink extends Component
{
    public function __construct($href, $active = null)
    {
        $this->href = $href;
        $this->active = $active ?? $href;
    }

    public function render(): View
    {
        $classes = ['font-medium', 'py-2', 'text-primary' => $this->isActive()];

        return view('components.nav-link', [
            'class' => Arr::toCssClasses($classes);
        ]);
    }

    protected function isActive(): bool
    {
        if (is_bool($this->active)) {
            return $this->active;
        }

        if (request()->is($this->active)) {
            return true;
        }

        if (request()->fullUrlIs($this->active)) {
            return true;
        }

        return request()->routeIs($this->active);
    }
}
```

```
<a href="{{ $href }}" {{ $attributes->class($class) }}>
    {{ $slot }}
</a>
```

```
<x-nav-link :href="route('projects.index')">Projects</x-nav-link>
<x-nav-link :href="route('projects.index')" active="projects.*">Projects</x-nav-link>
<x-nav-link :href="route('projects.index')" active="projects/*">Projects</x-nav-link>
<x-nav-link :href="route('projects.index')" :active="$tab = 'projects'">Projects</x-nav-link>
```

Tip given by @mpskovvang

Cleanup loops

Did you know the Blade `@each` directive can help cleanup loops in your templates?

```
// good
@foreach($item in $items)
    <div>
        <p>Name: {{ $item->name }}
        <p>Price: {{ $item->price }}
    </div>
@endforeach

// better (HTML extracted into partial)
@each('partials.item', $items, 'item')
```

Tip given by @kirschbaum_dev

Simple way to tidy up your Blade views

A simple way to tidy up your Blade views!
Use the `forelse loop`, instead of a `foreach loop` nested in an if statement

```
<!-- if/loop combination -->
@if ($orders->count())
    @foreach($orders as $order)
        <div>
            {{ $order->id }}
        </div>
    @endforeach
@else
    <p>You haven't placed any orders yet.</p>
@endif

<!-- Forelse alternative -->
@forelse($orders as $order)
    <div>
        {{ $order->id }}
    </div>
@empty
    <p>You haven't placed any orders yet.</p>
@endforelse
```

Tip given by @alexjgarrett

Routing

Route group within a group

In Routes, you can create a group within a group, assigning a certain middleware only to some URLs in the "parent" group.

```
Route::group(['prefix' => 'account', 'as' => 'account.'], function() {
    Route::get('login', 'AccountController@login');
    Route::get('register', 'AccountController@register');

    Route::group(['middleware' => 'auth'], function() {
        Route::get('edit', 'AccountController@edit');
    });
});
```

Wildcard subdomains

You can create route group by dynamic subdomain name, and pass its value to every route.

```
Route::domain('{username}.workspace.com')->group(function () {
    Route::get('user/{id}', function ($username, $id) {
        //
    });
});
```

What's behind the routes?

If you use Laravel UI package, you likely want to know what routes are actually behind `Auth::routes()`?

You can check the file `/vendor/laravel/ui/src/AuthRouteMethods.php`.

```
public function auth()
{
    return function ($options = []) {
        // Authentication Routes...
        $this->get('login', 'Auth\LoginController@showLoginForm')->name('login');
        $this->post('login', 'Auth\LoginController@login');
        $this->post('logout', 'Auth\LoginController@logout')->name('logout');
        // Registration Routes...
        if ($options['register'] ?? true) {
            $this->get('register', 'Auth\RegisterController@showRegistrationForm')->name('register');
            $this->post('register', 'Auth\RegisterController@register');
        }
        // Password Reset Routes...
        if ($options['reset'] ?? true) {
            $this->resetPassword();
        }
        // Password Confirmation Routes...
        if ($options['confirm'] ?? class_exists($this->prependGroupNamespace('Auth\ConfirmPasswordController'))) {
            $this->confirmPassword();
        }
        // Email Verification Routes...
        if ($options['verify'] ?? false) {
            $this->emailVerification();
        }
    };
}
```

The default use of that function is simply this:

```
Auth::routes(); // no parameters
```

But you can provide parameters to enable or disable certain routes:

```
Auth::routes([
    'login'    => true,
    'logout'   => true,
    'register' => true,
    'reset'    => true,  // for resetting passwords
    'confirm'  => false, // for additional password confirmations
    'verify'   => false, // for email verification
]);
```

Tip is based on suggestion by MimisK13

Route Model Binding: You can define a key

You can do Route model binding like `Route::get('api/users/{user}', function (App\User $user) { … }` - but not only by ID field. If you want `{user}` to be a `username` field, put this in the model:

```
public function getRouteKeyName() {
    return 'username';
}
```

Quickly Navigate from Routes file to Controller

This thing was optional before Laravel 8, and became a standard main syntax of routing in Laravel 8.

Instead of routing like this:

```
Route::get('page', 'PageController@action');
```

You can specify the Controller as a class:

```
Route::get('page', [\App\Http\Controllers\PageController::class, 'action']);
```

Then you will be able to click on **PageController** in PhpStorm, and navigate directly to Controller, instead of searching for it manually.

Or, to make it shorter, add this to top of Routes file:

```
use App\Http\Controllers\PageController;

// Then:
Route::get('page', [PageController::class, 'action']);
```

Route Fallback: When no Other Route is Matched

If you want to specify additional logic for not-found routes, instead of just throwing default 404 page, you may create a special Route for that, at the very end of your Routes file.

```
Route::group(['middleware' => ['auth'], 'prefix' => 'admin', 'as' => 'admin.'], function () {
    Route::get('/home', 'HomeController@index');
    Route::resource('tasks', 'Admin\TasksController');
});

// Some more routes....
Route::fallback(function() {
    return 'Hm, why did you land here somehow?';
});
```

Route Parameters Validation with RegExp

We can validate parameters directly in the route, with “where” parameter. A pretty typical case is to prefix your routes by language locale, like `fr/blog` and `en/article/333`. How do we ensure that those two first letters are not used for some other than language?

`routes/web.php`:

```
Route::group([
    'prefix' => '{locale}',
    'where' => ['locale' => '[a-zA-Z]{2}']
], function () {
    Route::get('/', 'HomeController@index');
    Route::get('article/{id}', 'ArticleController@show');
});
```

Rate Limiting: Global and for Guests/Users

You can limit some URL to be called a maximum of 60 times per minute, with `throttle:60,1`:

```
Route::middleware('auth:api', 'throttle:60,1')->group(function () {
    Route::get('/user', function () {
        //
    });
});
```

But also, you can do it separately for public and for logged-in users:

```
// maximum of 10 requests for guests, 60 for authenticated users
Route::middleware('throttle:10|60,1')->group(function () {
    //
});
```

Also, you can have a DB field users.rate_limit and limit the amount for specific user:

```
Route::middleware('auth:api', 'throttle:rate_limit,1')->group(function () {
    Route::get('/user', function () {
        //
    });
});
```

Query string parameters to Routes

If you pass additional parameters to the route, in the array, those key / value pairs will automatically be added to the generated URL's query string.

```
Route::get('user/{id}/profile', function ($id) {
    //
})->name('profile');

$url = route('profile', ['id' => 1, 'photos' => 'yes']); // Result: /user/1/profile?photos=yes
```

Separate Routes by Files

If you have a set of routes related to a certain "section", you may separate them in a special `routes/XXXXX.php` file, and just include it in `routes/web.php`

Example with `routes/auth.php` in Laravel Breeze by Taylor Otwell himself:

```
Route::get('/', function () {
    return view('welcome');
});

Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth'])->name('dashboard');

require __DIR__.'/auth.php';
```

Then, in `routes/auth.php`:

```
use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\RegisteredUserController;
// ... more controllers

use Illuminate\Support\Facades\Route;

Route::get('/register', [RegisteredUserController::class, 'create'])
                ->middleware('guest')
                ->name('register');

Route::post('/register', [RegisteredUserController::class, 'store'])
                ->middleware('guest');

// ... A dozen more routes
```

But you should use this `include()` only when that separate route file has the same settings for prefix/middlewares, otherwise it's better to group them in `app/Providers/RouteServiceProvider`:

```
public function boot()
{
    $this->configureRateLimiting();

    $this->routes(function () {
        Route::prefix('api')
            ->middleware('api')
            ->namespace($this->namespace)
            ->group(base_path('routes/api.php'));

        Route::middleware('web')
            ->namespace($this->namespace)
            ->group(base_path('routes/web.php'));

        // ... Your routes file listed next here
    });
}
```

Translate Resource Verbs

If you use resource controllers, but want to change URL verbs to non-English for SEO purposes, so instead of `/create` you want Spanish `/crear`, you can configure it by using `Route::resourceVerbs()` method in `App\Providers\RouteServiceProvider`:

```
public function boot()
{
    Route::resourceVerbs([
        'create' => 'crear',
        'edit' => 'editar',
    ]);

    // ...
}
```

Custom Resource Route Names

When using Resource Controllers, in `routes/web.php` you can specify `->names()` parameter, so the URL prefix in the browser and the route name prefix you use all over Laravel project may be different.

```
Route::resource('p', ProductController::class)->names('products');
```

So this code above will generate URLs like `/p`, `/p/{id}`, `/p/{id}/edit`, etc. But you would call them in the code by `route('products.index')`, `route('products.create')`, etc.

More Readable Route List

Have you ever run "php artisan route:list" and then realized that the list takes too much space and hard to read?

Here's the solution: `php artisan route:list --compact`

Then it shows 3 columns instead of 6 columns: shows only Method / URI / Action.

```
+----------+---------------------------------+-------------------------------------------------------------------------+
| Method   | URI                             | Action                                                                  |
+----------+---------------------------------+-------------------------------------------------------------------------+
| GET|HEAD | /                               | Closure                                                                 |
| GET|HEAD | api/user                        | Closure                                                                 |
| POST     | confirm-password                | App\Http\Controllers\Auth\ConfirmablePasswordController@store           |
| GET|HEAD | confirm-password                | App\Http\Controllers\Auth\ConfirmablePasswordController@show            |
| GET|HEAD | dashboard                       | Closure                                                                 |
| POST     | email/verification-notification | App\Http\Controllers\Auth\EmailVerificationNotificationController@store |
| POST     | forgot-password                 | App\Http\Controllers\Auth\PasswordResetLinkController@store             |
| GET|HEAD | forgot-password                 | App\Http\Controllers\Auth\PasswordResetLinkController@create            |
| POST     | login                           | App\Http\Controllers\Auth\AuthenticatedSessionController@store          |
| GET|HEAD | login                           | App\Http\Controllers\Auth\AuthenticatedSessionController@create         |
| POST     | logout                          | App\Http\Controllers\Auth\AuthenticatedSessionController@destroy        |
| POST     | register                        | App\Http\Controllers\Auth\RegisteredUserController@store                |
| GET|HEAD | register                        | App\Http\Controllers\Auth\RegisteredUserController@create               |
| POST     | reset-password                  | App\Http\Controllers\Auth\NewPasswordController@store                   |
| GET|HEAD | reset-password/{token}          | App\Http\Controllers\Auth\NewPasswordController@create                  |
| GET|HEAD | verify-email                    | App\Http\Controllers\Auth\EmailVerificationPromptController@__invoke    |
| GET|HEAD | verify-email/{id}/{hash}        | App\Http\Controllers\Auth\VerifyEmailController@__invoke                |
+----------+---------------------------------+-------------------------------------------------------------------------+
```

You can also specify the exact columns you want:

`php artisan route:list --columns=Method,URI,Name`

```
+----------+---------------------------------+---------------------+
| Method   | URI                             | Name                |
+----------+---------------------------------+---------------------+
| GET|HEAD | /                               |                     |
| GET|HEAD | api/user                        |                     |
| POST     | confirm-password                |                     |
| GET|HEAD | confirm-password                | password.confirm    |
| GET|HEAD | dashboard                       | dashboard           |
| POST     | email/verification-notification | verification.send   |
| POST     | forgot-password                 | password.email      |
| GET|HEAD | forgot-password                 | password.request    |
| POST     | login                           |                     |
| GET|HEAD | login                           | login               |
| POST     | logout                          | logout              |
| POST     | register                        |                     |
| GET|HEAD | register                        | register            |
| POST     | reset-password                  | password.update     |
| GET|HEAD | reset-password/{token}          | password.reset      |
| GET|HEAD | verify-email                    | verification.notice |
| GET|HEAD | verify-email/{id}/{hash}        | verification.verify |
+----------+---------------------------------+---------------------+
```

Eager load relationship

If you use Route Model Binding and think you can't use Eager Loading for relationships, think again.
So you use Route Model Binding

```
public function show(Product $product) {
    //
}
```

But you have a belongsTo relationship, and cannot use $product->with('category') eager loading?
You actually can! Load the relationship with `->load()`

```
public function show(Product $product) {
    $product->load('category');
    //
}
```

Localizing Resource URIs

If you use resource controllers, but want to change URL verbs to non-English, so instead of `/create` you want Spanish `/crear`, you can configure it with `Route::resourceVerbs()` method.

```
public function boot()
{
    Route::resourceVerbs([
        'create' => 'crear',
        'edit' => 'editar',
    ]);
    //
}
```

Resource Controllers naming

In Resource Controllers, in `routes/web.php` you can specify `->names()` parameter, so the URL prefix and the route name prefix may be different.
This will generate URLs like `/p`, `/p/{id}`, `/p/{id}/edit` etc. But you would call them: - route('products.index) - route('products.create) - etc

```
Route::resource('p', \App\Http\Controllers\ProductController::class)->names('products');
```

Easily highlight your navbar menus

Use `Route::is('route-name')` to easily highlight your navbar menus

```
<ul>
    <li @if(Route::is('home')) class="active" @endif>
        <a href="/">Home</a>
    </li>
    <li @if(Route::is('contact-us')) class="active" @endif>
        <a href="/contact-us">Contact us</a>
    </li>
</ul>
```

Tip given by @anwar_nairi

Generate absolute path using route() helper

```
route('page.show', $page->id);
// http://laravel.test/pages/1

route('page.show', $page->id, false);
// /pages/1
```

Tip given by @oliverds_

Override the route binding resolver for each of your models

You can override the route binding resolver for each of your models. In this example, I have no control over the @ sign in the URL, so using the `resolveRouteBinding` method, I'm able to remove the @ sign and resolve the model.

```
// Route
Route::get('{product:slug}', Controller::class);

// Request
https://nodejs.pub/@unlock/hello-world

// Product Model
public function resolveRouteBinding($value, $field = null)
{
    $value = str_replace('@', '', $value);

    return parent::resolveRouteBinding($value, $field);
}
```

Tip given by @Philo01

If you need public URL but you want them to be secured

If you need public URL but you want them to be secured, use Laravel signed URL

```
class AccountController extends Controller
{
    public function destroy(Request $request)
    {
        $confirmDeleteUrl = URL::signedRoute('confirm-destroy', [
            $user => $request->user()
        ]);
        // Send link by email...
    }

    public function confirmDestroy(Request $request, User $user)
    {
        if (! $request->hasValidSignature()) {
            abort(403);
        }

        // User confirmed by clikcing on the email
        $user->delete();

        return redirect()->route('home');
    }
}
```

Tip given by @anwar_nairi

Using Gate in middleware method

You can use the gates you specified in `App\Providers\AuthServiceProvider` in middleware method.

To do this, you just need to put inside the `can:` and the names of the necessary gates.

```
Route::put('/post/{post}', function (Post $post) {
    // The current user may update the post...
})->middleware('can:update,post');
```

Simple route with arrow function

You can use php arrow function in routing, without having to use anonymous function.

To do this, you can use `fn() =>`, it looks easier.

```
// Instead of
Route::get('/example', function () {
    return User::all();
});

// You can
Route::get('/example', fn () => User::all());
```

Validation

Image validation

While validating uploaded images, you can specify the dimensions you require.

```
['photo' => 'dimensions:max_width=4096,max_height=4096']
```

Custom validation error messages

You can customize validation error messages per **field**, **rule** and **language** - just create a specific language file `resources/lang/xx/validation.php` with appropriate array structure.

```
'custom' => [
    'email' => [
       'required' => 'We need to know your e-mail address!',
    ],
],
```

Validate dates with "now" or "yesterday" words

You can validate dates by rules before/after and passing various strings as a parameter, like: `tomorrow`, `now`, `yesterday`. Example: `'start_date' => 'after:now'`. It's using strtotime() under the hood.

```
$rules = [
    'start_date' => 'after:tomorrow',
    'end_date' => 'after:start_date'
];
```

Validation Rule with Some Conditions

If your validation rules depend on some condition, you can modify the rules by adding `withValidator()` to your `FormRequest` class, and specify your custom logic there. Like, if you want to add validation rule only for some user role.

```
use Illuminate\Validation\Validator;
class StoreBlogCategoryRequest extends FormRequest {
    public function withValidator(Validator $validator) {
        if (auth()->user()->is_admin) {
            $validator->addRules(['some_secret_password' => 'required']);
        }
    }
}
```

Change Default Validation Messages

If you want to change default validation error message for specific field and specific validation rule, just add a `messages()` method into your `FormRequest` class.

```
class StoreUserRequest extends FormRequest
{
    public function rules()
    {
        return ['name' => 'required'];
    }

    public function messages()
    {
        return ['name.required' => 'User name should be real name'];
    }
}
```

Prepare for Validation

If you want to modify some field before default Laravel validation, or, in other words, "prepare" that field, guess what - there's a method `prepareForValidation()` in `FormRequest` class:

```
protected function prepareForValidation()
{
    $this->merge([
        'slug' => Illuminate\Support\Str::slug($this->slug),
    ]);
}
```

Stop on First Validation Error

By default, Laravel validation errors will be returned in a list, checking all validation rules. But if you want the process to stop after the first error, use validation rule called `bail`:

```
$request->validate([
    'title' => 'bail|required|unique:posts|max:255',
    'body' => 'required',
]);
```

If you need to stop validation on the first error in `FormRequest` class, you can set `stopOnFirstFailure` property to `true`:

```
protected $stopOnFirstFailure = true;
```

Throw 422 status code without using validate() or Form Request

If you don't use validate() or Form Request, but still need to throw errors with the same 422 status code and error structure, you can do it manually `throw ValidationException::withMessages()`

```
if (! $user || ! Hash::check($request->password, $user->password)) {
    throw ValidationException::withMessages([
        'email' => ['The provided credentials are incorrect.'],
    ]);
}
```

Rules depending on some other conditions

If your rules are dynamic and depend on some other condition, you can create that array of rules on the fly

```
    public function store(Request $request)
    {
        $validationArray = [
            'title' => 'required',
            'company' => 'required',
            'logo' => 'file|max:2048',
            'location' => 'required',
            'apply_link' => 'required|url',
            'content' => 'required',
            'payment_method_id' => 'required'
        ];

        if (!Auth::check()) {
            $validationArray = array_merge($validationArray, [
                'email' => 'required|email|unique:users',
                'password' => 'required|confirmed|min:5',
                'name' => 'required'
            ]);
        }
        //
    }
```

With Rule::when() we can conditionally apply validation rules

Thanks to Rule::when() we can conditionally apply validation rules in laravel.
In this example we validate the value of the vote only if the user can actually vote the post.

```
use Illuminate\Validation\Rule;

public function rules()
{
    return [
        'vote' => Rule::when($user->can('vote', $post), 'required|int|between:1,5'),
    ]
}
```

Tip given by @cerbero90

Use this property in the request classes to stop the validation of the whole request attributes

Use this property in the request classes to stop the validation of the whole request attributes.

Hint Direct
This is different from `Bail` rule that stops the validation for just a single attribute if one of its rules doesn't validate.

```
/**
* Indicated if the validator should stop
 * the entire validation once a single
 * rule failure has occurred.
 */
protected $stopOnFirstFailure = true;
```

Tip given by @Sala7JR

Rule::unique doesn't take into the SoftDeletes Global Scope applied on the Model

Strange that `Rule::unique` doesn't take into the SoftDeletes Global Scope applied on the Model, by default.
But `withoutTrashed()` method is available

```
Rule::unique('users', 'email')->withoutTrashed();
```

Tip given by @Zubairmohsin33

Validator::sometimes() method allows us to define when a validation rule should be applied

The laravel `Validator::sometimes()` method allows us to define when a validation rule should be applied, based on the input provided.
The snippet shows how to prohibit the use of a coupon if the quantity of the purchased items is not enough.

```
$data = [
    'coupon' => 'PIZZA_PARTY',
    'items' => [
        [
            'id' => 1,
            'quantity' => 2
        ],
        [
            'id' => 2,
            'quantity' => 2,
        ],
    ],
];

$validator = Validator::make($data, [
    'coupon' => 'exists:coupons,name',
    'items' => 'required|array',
    'items.*.id' => 'required|int',
    'items.*.quantity' => 'required|int',
]);

$validator->sometimes('coupon', 'prohibited', function (Fluent $data) {
    return collect($data->items)->sum('quantity') < 5;
});

// throws a ValidationException as the quantity provided is not enough
$validator->validate();
```

Tip given by @cerbero90

Array elements validation

If you want to validate elements of an array that you submited use dot notation in rules with '*'

```
// say you have this array
// array in request 'user_info'
$request->validated()->user_info = [
    [
        'name' => 'Qasim',
        'age' => 26,
    ],
    [
        'name' => 'Ahmed',
        'age' => 23,
    ],
];

// Rule
$rules = [
    'user_info.*.name' => ['required', 'alpha'],
    'user_info.*.age' => ['required', 'numeric'],
];
```

Tip given by HydroMoon

Password::defaults method

You can enforce specific rules when validating user-supplied passwords by using the Password::defaults method. It includes options for requiring letters, numbers, symbols, and more.

```
class AppServiceProvider
{
    public function boot(): void
    {
        Password::defaults(function () {
            return Password::min(12)
                ->letters()
                ->numbers()
                ->symbols()
                ->mixedCase()
                ->uncompromised();
        })
    }
}

request()->validate([
    ['password' => ['required', Password::defaults()]]
])
```

Tip given by @mattkingshott

Form Requests for validation redirection

when using Form Requests for validation, by default the validation error will redirect back to the previous page, but you can override it.
Just define the property of `$redirect` or `$redirectRoute`.
Link to docs

```
// The URI that users should be redirected to if validation fails./
protected $redirect = '/dashboard';

// The route that users should be redirected to if validation fails.
protected $redirectRoute = 'dashboard';
```

Mac validation rule

New mac_address validation rule added in Laravel 8.77

```
$trans = $this->getIlluminateArrayTranslator();
$validator = new Validator($trans, ['mac' => '01-23-45-67-89-ab'], ['mac' => 'mac_address']);
$this->assertTrue($validator->passes());
```

Tip given by @Teacoders

Collections

Don’t Filter by NULL in Collections

You can filter by NULL in Eloquent, but if you're filtering the **collection** further - filter by empty string, there's no "null" in that field anymore.

```
// This works
$messages = Message::where('read_at is null')->get();

// Won’t work - will return 0 messages
$messages = Message::all();
$unread_messages = $messages->where('read_at is null')->count();

// Will work
$unread_messages = $messages->where('read_at', '')->count();
```

Use groupBy on Collections with Custom Callback Function

If you want to group result by some condition which isn’t a direct column in your database, you can do that by providing a closure function.

For example, if you want to group users by day of registration, here’s the code:

```
$users = User::all()->groupBy(function($item) {
    return $item->created_at->format('Y-m-d');
});
```

⚠ Notice: it is done on a `Collection` class, so performed **AFTER** the results are fetched from the database.

Multiple Collection Methods in a Row

If you query all results with `->all()` or `->get()`, you may then perform various Collection operations on the same result, it won’t query database every time.

```
$users = User::all();
echo 'Max ID: ' . $users->max('id');
echo 'Average age: ' . $users->avg('age');
echo 'Total budget: ' . $users->sum('budget');
```

Calculate Sum with Pagination

How to calculate the sum of all records when you have only the PAGINATED collection? Do the calculation BEFORE the pagination, but from the same query.

```
// How to get sum of post_views with pagination?
$posts = Post::paginate(10);
// This will be only for page 1, not ALL posts
$sum = $posts->sum('post_views');

// Do this with Query Builder
$query = Post::query();
// Calculate sum
$sum = $query->sum('post_views');
// And then do the pagination from the same query
$posts = $query->paginate(10);
```

Serial no in foreach loop with pagination

We can use foreach collection items index as serial no (SL) in pagination.

```
...
<th>Serial</th>
...
@foreach ($products as $product)
<tr>
    <td>{{ $loop->index + $product->firstItem() }}</td>
    ...
@endforeach
```

it will solve the issue of next pages(?page=2&...) index count from continue.

Higher order collection methods

Collections have higher order methods, this are methods that can be chained , like `groupBy()` , `map()` ... Giving you a fluid syntax. This example calculates the price per group of products on an offer.

```
$offer = [
        'name'  => 'offer1',
        'lines' => [
            ['group' => 1, 'price' => 10],
            ['group' => 1, 'price' => 20],
            ['group' => 2, 'price' => 30],
            ['group' => 2, 'price' => 40],
            ['group' => 3, 'price' => 50],
            ['group' => 3, 'price' => 60]
        ]
];

$totalPerGroup = collect($offer->lines)->groupBy('group')->map(fn($group) => $group->sum('price'));
```

Auth

Check Multiple Permissions at Once

In addition to `@can` Blade directive, did you know you can check multiple permissions at once with `@canany` directive?

```
@canany(['update', 'view', 'delete'], $post)
    // The current user can update, view, or delete the post
@elsecanany(['create'], \App\Post::class)
    // The current user can create a post
@endcanany
```

More Events on User Registration

Want to perform some actions after new user registration? Head to `app/Providers/EventServiceProvider.php` and add more Listeners classes, and then in those classes implement `handle()` method with `$event->user` object

```
class EventServiceProvider extends ServiceProvider
{
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,

            // You can add any Listener class here
            // With handle() method inside of that class
        ],
    ];
```

Did you know about Auth::once()?

You can login with user only for ONE REQUEST, using method `Auth::once()` .
No sessions or cookies will be utilized, which means this method may be helpful when building a stateless API.

```
if (Auth::once($credentials)) {
    //
}
```

Change API Token on users password update

It's convenient to change the user's API Token when its password changes.

Model:

```
public function setPasswordAttribute($value)
{
    $this->attributes['password'] = $value;
    $this->attributes['api_token'] = Str::random(100);
}
```

Override Permissions for Super Admin

If you've defined your Gates but want to override all permissions for SUPER ADMIN user, to give that superadmin ALL permissions, you can intercept gates with `Gate::before()` statement, in `AuthServiceProvider.php` file.

```
// Intercept any Gate and check if it's super admin
Gate::before(function($user, $ability) {
    if ($user->is_super_admin == 1) {
        return true;
    }
});

// Or if you use some permissions package...
Gate::before(function($user, $ability) {
    if ($user->hasPermission('root')) {
        return true;
    }
});
```

Mail

Testing email into laravel.log

If you want to test email contents in your app but unable or unwilling to set up something like Mailgun, use `.env` parameter `MAIL_DRIVER=log` and all the email will be saved into `storage/logs/laravel.log` file, instead of actually being sent.

Preview Mailables

If you use Mailables to send email, you can preview the result without sending, directly in your browser. Just return a Mailable as route result:

```
Route::get('/mailable', function () {
    $invoice = App\Invoice::find(1);
    return new App\Mail\InvoicePaid($invoice);
});
```

Preview Mail without Mailables

You can also preview your email without Mailables. For instance, when you are creating notification, you can specify the markdown that may be use for your mail notification.

```
use Illuminate\Notifications\Messages\MailMessage;

Route::get('/mailable', function () {
    $invoice = App\Invoice::find(1);
    return (new MailMessage)->markdown('emails.invoice-paid', compact('invoice'));
});
```

You may also use other methods provided by `MailMessage` object such as `view` and others.

Tip given by @raditzfarhan

Default Email Subject in Laravel Notifications

If you send Laravel Notification and don't specify subject in **toMail()**, default subject is your notification class name, CamelCased into Spaces.

So, if you have:

```
class UserRegistrationEmail extends Notification {
    //
}
```

Then you will receive an email with subject **User Registration Email**.

Send Notifications to Anyone

You can send Laravel Notifications not only to a certain user with `$user->notify()`, but also to anyone you want, via `Notification::route()`, with so-called "on-demand" notifications:

```
Notification::route('mail', 'taylor@example.com')
        ->route('nexmo', '5555555555')
        ->route('slack', 'https://hooks.slack.com/services/...')
        ->notify(new InvoicePaid($invoice));
```

Artisan

Artisan command parameters

When creating Artisan command, you can ask the input in variety of ways: `$this->confirm()`, `$this->anticipate()`, `$this->choice()`.

```
// Yes or no?
if ($this->confirm('Do you wish to continue?')) {
    //
}

// Open question with auto-complete options
$name = $this->anticipate('What is your name?', ['Taylor', 'Dayle']);

// One of the listed options with default index
$name = $this->choice('What is your name?', ['Taylor', 'Dayle'], $defaultIndex);
```

Maintenance Mode

If you want to enable maintenance mode on your page, execute the down Artisan command:

```
php artisan down
```

Then people would see default 503 status page.

You may also provide flags, in Laravel 8: - the path the user should be redirected to - the view that should be prerendered - secret phrase to bypass maintenance mode - status code during maintenance mode - retry page reload every X seconds

```
php artisan down --redirect="/" --render="errors::503" --secret="1630542a-246b-4b66-afa1-dd72a4c43515" --status=200 --retry=60
```

Before Laravel 8: - message that would be shown - retry page reload every X seconds - still allow the access to some IP address

```
php artisan down --message="Upgrading Database" --retry=60 --allow=127.0.0.1
```

When you've done the maintenance work, just run:

```
php artisan up
```

Artisan command help

To check the options of artisan command, Run artisan commands with `--help` flag. For example, `php artisan make:model --help` and see how many options

you have:

```
Options:
  -a, --all             Generate a migration, seeder, factory, and resource controller for the model
  -c, --controller      Create a new controller for the model
  -f, --factory         Create a new factory for the model
      --force           Create the class even if the model already exists
  -m, --migration       Create a new migration file for the model
  -s, --seed            Create a new seeder file for the model
  -p, --pivot           Indicates if the generated model should be a custom intermediate table model
  -r, --resource        Indicates if the generated controller should be a resource controller
      --api             Indicates if the generated controller should be an API controller
  -h, --help            Display this help message
  -q, --quiet           Do not output any message
  -V, --version         Display this application version
      --ansi            Force ANSI output
      --no-ansi         Disable ANSI output
  -n, --no-interaction  Do not ask any interactive question
      --env[=ENV]       The environment the command should run under
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
```

Exact Laravel version

Find out exactly what Laravel version you have in your app, by running command `php artisan --version`

Launch Artisan command from anywhere

If you have an Artisan command, you can launch it not only from Terminal, but also from anywhere in your code, with parameters. Use Artisan::call() method:

```
Route::get('/foo', function () {
    $exitCode = Artisan::call('email:send', [
        'user' => 1, '--queue' => 'default'
    ]);

    //
});
```

Factories

Factory callbacks

While using factories for seeding data, you can provide Factory Callback functions to perform some action after record is inserted.

```
$factory->afterCreating(App\User::class, function ($user, $faker) {
    $user->accounts()->save(factory(App\Account::class)->make());
});
```

Generate Images with Seeds/Factories

Did you know that Faker can generate not only text values but also IMAGES? See `avatar` field here - it will generate 50x50 image:

```
$factory->define(User::class, function (Faker $faker) {
    return [
        'name' => $faker->name,
        'email' => $faker->unique()->safeEmail,
        'email_verified_at' => now(),
        'password' => bcrypt('password'),
        'remember_token' => Str::random(10),
        'avatar' => $faker->image(storage_path('images'), 50, 50)
    ];
});
```

Override values and apply custom login to them

When creating records with Factories, you can use Sequence class to override some values and apply custom logic to them.

```
$users = User::factory()
                ->count(10)
                ->state(new Sequence(
                    ['admin' => 'Y'],
                    ['admin' => 'N'],
                ))
                ->create();
```

Using factories with relationships

When using factories with relationships, Laravel also provides magic methods.

```
// magic factory relationship methods
User::factory()->hasPosts(3)->create();

// instead of
User::factory()->has(Post::factory()->count(3))->create();
```

Tip given by @oliverds_

Create models without dispatching any events

Sometimes you may wish to `update` a given model without dispatching any events. You may accomplish this using the `updateQuietly` method

```
Post::factory()->createOneQuietly();

Post::factory()->count(3)->createQuietly();

Post::factory()->createManyQuietly([
    ['message' => 'A new comment'],
    ['message' => 'Another new comment'],
]);
```

Useful for() method

The Laravel factory has a very useful `for()` method. You can use it to create `belongsTo()` relationships.

```
public function run()
{
    Product::factory()
        ->count(3);
        ->for(Category::factory()->create())
        ->create();
}
```

Tip given by @mmartin_joo

Log and debug

Logging with parameters

You can write `Log::info()`, or shorter `info()` message with additional parameters, for more context about what happened.

```
Log::info('User failed to login.', ['id' => $user->id]);
```

More convenient DD

Instead of doing `dd($result)` you can put `->dd()` as a method directly at the end of your Eloquent sentence, or any Collection.

```
// Instead of
$users = User::where('name', 'Taylor')->get();
dd($users);
// Do this
$users = User::where('name', 'Taylor')->get()->dd();
```

Log with context

New in Laravel 8.49: `Log::withContext()` will help you to differentiate the Log messages between different requests.
If you create a Middleware and set this context, all Log messages will contain that context, and you'll be able to search them easier.

```
public function handle(Request $request, Closure $next)
{
    $requestId = (string) Str::uuid();

    Log::withContext(['request-id' => $requestId]);

    $response = $next($request);

    $response->header('request-id', $requestId);

    return $response;
}
```

Quickly output an Eloquent query in its SQL form

If you want to quickly output an Eloquent query in its SQL form, you can invoke the toSql() method onto it like so

```
$invoices = Invoice::where('client', 'James pay')->toSql();

dd($invoices)
// select * from `invoices` where `client` = ?
```

Tip given by @devThaer

Log all the database queries during development

If you want to log all the database queries during development add this snippet to your AppServiceProvider

```
public function boot()
{
    if (App::environment('local')) {
        DB::listen(function ($query) {
            logger(Str::replaceArray('?', $query->bindings, $query->sql));
        });
    }
}
```

Tip given by @mmartin_joo

API

API Resources: With or Without "data"?

If you use Eloquent API Resources to return data, they will be automatically wrapped in 'data'. If you want to remove it, add `JsonResource::withoutWrapping();` in `app/Providers/AppServiceProvider.php`.

```
class AppServiceProvider extends ServiceProvider
{
    public function boot()
    {
        JsonResource::withoutWrapping();
    }
}
```

Tip given by @phillipmwaniki

API Return "Everything went ok"

If you have API endpoint which performs some operations but has no response, so you wanna return just "everything went ok", you may return 204 status code "No content". In Laravel, it's easy: `return response()->noContent();`.

```
public function reorder(Request $request)
{
    foreach ($request->input('rows', []) as $row) {
        Country::find($row['id'])->update(['position' => $row['position']]);
    }

    return response()->noContent();
}
```

Avoid N+1 queries in API resources

You can avoid N+1 queries in API resources by using the `whenLoaded()` method.
This will only append the department if it's already loaded in the Employee model.
Without `whenLoaded()` there is always a query for the department

```
class EmplyeeResource extends JsonResource
{
    public function toArray($request): array
    {
        return [
            'id' => $this->uuid,
            'fullName' => $this->full_name,
            'email' => $this->email,
            'jobTitle' => $this->job_title,
            'department' => DepartmentResource::make($this->whenLoaded('department')),
        ];
    }
}
```

Tip given by @mmartin_joo

Other

- Add Parameters to Pagination Links
- Repeatable Callback Functions
- Request: has any
- Simple Pagination
- Data Get Function
- Blade directive to add true/false conditions
- Jobs can be used without queues
- Use faker outside factories or seeders
- Schedule things
- Search Laravel docs
- Filter route:list
- Blade directive for not repeating yourself
- Artisan commands help
- Disable lazy loading when running your tests
- Using two amazing helpers in Laravel will bring magic results
- Request parameter default value
- Pass middleware directly into the route without register it
- Transforming an array to CssClasses
- "upcomingInvoice" method in Laravel Cashier (Stripe)
- Laravel Request exists() vs has()
- There are multiple ways to return a view with variables
- Schedule regular shell commands
- HTTP client request without verifying
- Test that doesn't assert anything
- "Str::mask()" method
- Extending Laravel classes
- Can feature
- Temporary download URLs
- Dealing with deeply-nested arrays
- Customize how your exceptions are rendered
- The tap helper
- Reset all of the remaining time units
- Scheduled commands in the console kernel can automatically email their output if something goes wrong
- Be careful when constructing your custom filtered queries using GET parameters
- Dust out your bloated route file
- You can send e-mails to a custom log file
- Markdown made easy
- Simplify if on a request with whenFilled() helper
- Pass arguments to middleware
- Get value from session and forget
- $request->date() method
- Use through instead of map when using pagination

Localhost in .env

Don't forget to change `APP_URL` in your `.env` file from `http://localhost` to the real URL, cause it will be the basis for any links in your email notifications and elsewhere.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:9PHz3TL5C4YrdV6Gg/Xkkmx9btaE93j7rQTUZWm2MqU=
APP_DEBUG=true
APP_URL=http://localhost
```

When (NOT) to run "composer update"

Not so much about Laravel, but... Never run `composer update` on production live server, it's slow and will "break" repository. Always run `composer update` locally on your computer, commit new `composer.lock` to the repository, and run `composer install` on the live server.

Composer: Check for Newer Versions

If you want to find out which of your `composer.json` packages have released newer versions, just run `composer outdated`. You will get a full list with all information, like this below.

```
phpdocumentor/type-resolver 0.4.0 0.7.1
phpunit/php-code-coverage   6.1.4 7.0.3 Library that provides collection, processing, and rende...
phpunit/phpunit             7.5.9 8.1.3 The PHP Unit Testing framework.
ralouphie/getallheaders     2.0.5 3.0.3 A polyfill for getallheaders.
sebastian/global-state      2.0.0 3.0.0 Snapshotting of global state
```

Auto-Capitalize Translations

In translation files (`resources/lang`), you can specify variables not only as `:variable` , but also capitalized as `:VARIABLE` or `:Variable` - and then whatever value you pass - will be also capitalized automatically.

```
// resources/lang/en/messages.php
'welcome' => 'Welcome, :Name'

// Result: "Welcome, Taylor"
echo __('messages.welcome', ['name' => 'taylor']);
```

Carbon with Only Hours

If you want to have a current date without seconds and/or minutes, use Carbon's methods like `setSeconds(0)` or `setMinutes(0)`.

```
// 2020-04-20 08:12:34
echo now();

// 2020-04-20 08:12:00
echo now()->setSeconds(0);

// 2020-04-20 08:00:00
echo now()->setSeconds(0)->setMinutes(0);

// Another way - even shorter
echo now()->startOfHour();
```

Single Action Controllers

If you want to create a controller with just one action, you can use `__invoke()` method and even create "invokable" controller.

Route:

```
Route::get('user/{id}', 'ShowProfile');
```

Artisan:

```
php artisan make:controller ShowProfile --invokable
```

Controller:

```
class ShowProfile extends Controller
{
    public function __invoke($id)
    {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
```

Redirect to Specific Controller Method

You can `redirect()` not only to URL or specific route, but to a specific Controller's specific method, and even pass the parameters. Use this:

```
return redirect()->action('SomeController@method', ['param' => $value]);
```

Use Older Laravel Version

If you want to use OLDER version instead of the newest Laravel, use this command:

```
composer create-project --prefer-dist laravel/laravel project "7.*"
```

Change 7.* to whichever version you want.

Add Parameters to Pagination Links

In default Pagination links, you can pass additional parameters, preserve the original query string, or even point to a specific `#xxxxx` anchor.

```
{{ $users->appends(['sort' => 'votes'])->links() }}

{{ $users->withQueryString()->links() }}

{{ $users->fragment('foo')->links() }}
```

Repeatable Callback Functions

If you have a callback function that you need to re-use multiple times, you can assign it to a variable, and then re-use.

```
$userCondition = function ($query) {
    $query->where('user_id', auth()->id());
};

// Get articles that have comments from this user
// And return only those comments from this user
$articles = Article::with(['comments' => $userCondition])
    ->whereHas('comments', $userCondition)
    ->get();
```

Request: has any

You can check not only one parameter with `$request->has()` method, but also check for multiple parameters present, with `$request->hasAny()` :

```
public function store(Request $request)
{
    if ($request->hasAny(['api_key', 'token'])) {
        echo 'We have API key passed';
    } else {
        echo 'No authorization parameter';
    }
}
```

Simple Pagination

In pagination, if you want to have just "Previous/next" links instead of all the page numbers (and have fewer DB queries because of that), just change `paginate()` to `simplePaginate()` :

```
// Instead of
$users = User::paginate(10);

// You can do this
$users = User::simplePaginate(10);
```

Data Get Function

If you have an array complex data structure, for example a nested array with objects. You can use `data_get()` helper function retrieves a value from a nested array or object using "dot" notation and wildcard:

```
// We have an array
[
  0 =>
    ['user_id' =>'some user id', 'created_at' => 'some timestamp', 'product' => {object Product}, etc],
  1 =>
    ['user_id' =>'some user id', 'created_at' => 'some timestamp', 'product' => {object Product}, etc],
  2 =>  etc
]

// Now we want to get all products ids. We can do like this:

data_get($yourArray,  '*.product.id');

// Now we have all products ids [1, 2, 3, 4, 5, etc...]
```

Blade directive to add true/false conditions

New in Laravel 8.51: `@class` Blade directive to add true/false conditions on whether some CSS class should be added. Read more in docs
Before:

```
<div class="@if ($active) underline @endif">`
```

Now:

```
<div @class(['underline' => $active])>
```

```
@php
    $isActive = false;
    $hasError = true;
@endphp

<span @class([
    'p-4',
    'font-bold' => $isActive,
    'text-gray-500' => ! $isActive,
    'bg-red' => $hasError,
])></span>

<span class="p-4 text-gray-500 bg-red"></span>
```

Tip given by @Teacoders

Jobs can be used without queues

Jobs are discussed in the "Queues" section of the docs, but you can use Jobs without queues, just as classes to delegate tasks to. Just call `$this->dispatchNow()` from Controllers

```
public function approve(Article $article)
{
    //
    $this->dispatchNow(new ApproveArticle($article));
    //
}
```

Use faker outside factories or seeders

If you want to generate some fake data, you can use Faker even outside factories or seeds, in any class.
*Keep in mind: to use it in **production**, you need to move faker from `"require-dev"` to `"require"` in `composer.json`*

```
use Faker;

class WhateverController extends Controller
{
    public function whatever_method()
    {
        $faker = Faker\Factory::create();
        $address = $faker->streetAddress;
    }
}
```

Schedule things

You can schedule things to run daily/hourly in a lot of different structures.
You can schedule an artisan command, a Job class, an invokable class, a callback function, and even execute a shell script.

```
use App\Jobs\Heartbeat;

$schedule->job(new Heartbeat)->everyFiveMinutes();
```

```
$schedule->exec('node /home/forge/script.js')->daily();
```

```
use App\Console\Commands\SendEmailsCommand;

$schedule->command('emails:send Taylor --force')->daily();

$schedule->command(SendEmailsCommand::class, ['Taylor', '--force'])->daily();
```

```
protected function schedule(Schedule $schedule)
{
    $schedule->call(function () {
        DB::table('recent_users')->delete();
    })->daily();
}
```

Search Laravel docs

If you want to search Laravel Docs for some keyword, by default it gives you only the TOP 5 results. Maybe there are more?
If you want to see ALL results, you may go to the Github Laravel docs repository and search there directly. https://github.com/laravel/docs

Filter route:list

New in Laravel 8.34: `php artisan route:list` gets additional flag `--except-path`, so you would filter out the routes you don't want to see. [See original PR](New in Laravel 8.34: `php artisan route:list` gets additional flag `--except-path`, so you would filter out the routes you don't want to see. See original PR

Blade directive for not repeating yourself

If you keep doing the same formatting of the data in multiple Blade files, you may create your own Blade directive.
Here's an example of money amount formatting using the method from Laravel Cashier.

```
"require": {
        "laravel/cashier": "^12.9",
}
```

```
public function boot()
{
    Blade::directive('money', function ($expression) {
        return "<?php echo Laravel\Cashier\Cashier::formatAmount($expression, config('cashier.currency')); ?>";
    });
}
```

```
<div>Price: @money($book->price)</div>
@if($book->discount_price)
    <div>Discounted price: @money($book->dicount_price)</div>
@endif
```

Artisan commands help

If you are not sure about the parameters of some Artisan command, or you want to know what parameters are available, just type `php artisan help [a command you want]`.

Disable lazy loading when running your tests

If you don't want to prevent lazy loading when running your tests you can disable it

```
Model::preventLazyLoading(!$this->app->isProduction() && !$this->app->runningUnitTests());
```

Tip given by @djgeisi

Using two amazing helpers in Laravel will bring magic results

Using two amazing helpers in Laravel will bring magic results...
In this case, the service will be called and retried (retry). If it stills failing, it will be reported, but the request won't fail (rescue)

```
rescue(function () {
    retry(5, function () {
        $this->service->callSomething();
    }, 200);
});
```

Tip given by @JuanDMeGon

Request parameter default value

Here we are checking if there is a per_page (or any other parameter) value then we will use it, otherwise, we will use a default one.

```
// Isteand of this
$perPage = request()->per_page ? request()->per_page : 20;

// You can do this
$perPage = request('per_page', 20);
```

Tip given by @devThaer

Pass middleware directly into the route without register it

```
Route::get('posts', PostController::class)
    ->middleware(['auth', CustomMiddleware::class])
```

Tip given by @sky_0xs

Transforming an array to CssClasses

```
use Illuminate\Support\Arr;

$array = ['p-4', 'font-bold' => $isActive, 'bg-red' => $hasError];

$isActive = false;
$hasError = true;

$classes = Arr::toCssClasses($array);

/*
 * 'p-4 bg-red'
 */
```

Tip given by @dietsedev

"upcomingInvoice" method in Laravel Cashier (Stripe)

You can show how much a customer will pay in the next billing cycle.
There is a "upcomingInvoice" method in Laravel Cashier (Stripe) to get the upcoming invoice details.

```
Route::get('/profile/invoices', function (Request $request) {
    return view('/profile/invoices', [
        'upcomingInvoice' => $request->user()->upcomingInvoice(),
        'invoices' => $request-user()->invoices(),
    ]);
});
```

Tip given by @oliverds_

Laravel Request exists() vs has()

```
// https://example.com?popular
$request->exists('popular') // true
$request->has('popular') // false

// https://example.com?popular=foo
$request->exists('popular') // true
$request->has('popular') // true
```

Tip given by @coderahuljat

There are multiple ways to return a view with variables

```
// First way ->with()
return view('index')
    ->with('projects', $projects)
    ->with('tasks', $tasks)

// Second way - as an array
return view('index', [
        'projects' => $projects,
        'tasks' => $tasks
    ]);

// Third way - the same as second, but with variable
$data = [
    'projects' => $projects,
    'tasks' => $tasks
];
return view('index', $data);

// Fourth way - the shortest - compact()
return view('index', compact('projects', 'tasks'));
```

Schedule regular shell commands

We can schedule regular shell commands within Laravel scheduled command

```
// app/Console/Kernel.php

class Kernel extends ConsoleKernel
{
    protected function shedule(Schedule $shedule)
    {
        $shedule->exec('node /home/forge/script.js')->daily();
    }
}
```

Tip given by @anwar_nairi

HTTP client request without verifying

Sometimes, you may want to send HTTP request without verifying SSL in your local environment, you can do like so:

```
return Http::withoutVerifying()->post('https://example.com');
```

If you want to set multiple options, you can use `withOptions`.

```
return Http::withOptions([
    'verify' => false,
    'allow_redirects' => true
])->post('https://example.com');
```

Tip given by @raditzfarhan

Test that doesn't assert anything

Test that doesn't assert anything, just launch something which may or may not throw an exception

```
class MigrationsTest extends TestCase
{
    public function test_successful_foreign_key_in_migrations()
    {
        // We just test if the migrations succeeds or throws an exception
        $this->expectNotToPerformAssertions();


       Artisan::call('migrate:fresh', ['--path' => '/databse/migrations/task1']);
    }
}
```

"Str::mask()" method

Laravel 8.69 released with "Str::mask()" method which masks a portion of string with a repeated character

```
class PasswordResetLinkController extends Controller
{
    public function sendResetLinkResponse(Request $request)
    {
        $userEmail = User::where('email', $request->email)->value('email'); // username@domain.com

        $maskedEmail = Str::mask($userEmail, '*', 4); // user***************

        // If needed, you provide a negative number as the third argument to the mask method,
        // which will instruct the method to begin masking at the given distance from the end of the string

        $maskedEmail = Str::mask($userEmail, '*', -16, 6); // use******domain.com
    }
}
```

Tip given by @Teacoders

Extending Laravel classes

There is a method called macro on a lot of built-in Laravel classes. For example Collection, Str, Arr, Request, Cache, File, and so on.
You can define your own methods on these classes like this:

```
Str::macro('lowerSnake', function (string $str) {
    return Str::lower(Str::snake($str));
});

// Will return: "my-string"
Str::lowerSnake('MyString');
```

Tip given by @mmartin_joo

Can feature

If you are running Laravel `v8.70`, you can chain `can()` method directly instead of `middleware('can:..')`

```
// instead of
Route::get('users/{user}/edit', function (User $user) {
    ...
})->middleware('can:edit,user');

// you can do this
Route::get('users/{user}/edit', function (User $user) {
    ...
})->can('edit' 'user');

// PS: you must write UserPolicy to be able to do this in both cases
```

Tip given by @sky_0xs

Temporary download URLs

You can use temporary download URLs for your cloud storage resources to prevent unwanted access. For example, when a user wants to download a file, we redirect to an s3 resource but have the URL expire in 5 seconds.

```
public function download(File $file)
{
    // Initiate file download by redirecting to a temporary s3 URL that expires in 5 seconds
    return redirect()->to(
        Storage::disk('s3')->temporaryUrl($file->name, now()->addSeconds(5))
    );
}
```

Tip given by @Philo01

Dealing with deeply-nested arrays

Dealing with deeply-nested arrays can result in missing key / value exceptions. Fortunately, Laravel's data_get() helper makes this easy to avoid. It also supports deeply-nested objects.

Deeply-nested arrays are a nightmare when they may be missing properties that you need.
In the example below, if either `request`, `user` or `name` are missing then you'll get errors.

```
$value = $payload['request']['user']['name']
```

Instead, use the `data_get()` helper to access a deeply-nested array item using dot notation.

```
$value = data_get($payload, 'request.user.name');
```

We can also avoid any errors caused by missing properties by supplying a default value.

```
$value = data_get($payload, 'request.user.name', 'John');
```

Tip given by @mattkingshott

Customize how your exceptions are rendered

You can customize how your exceptions are rendered by adding a 'render' method to your exception.
For example, this allows you to return JSON instead of a Blade view when the request expects JSON.

```
abstract class BaseException extends Exception
{
    public function render(Request $request)
    {
        if ($request->expectsJson()) {
            return response()->json([
                'meta' => [
                    'valid'   => false,
                    'status'  => static::ID,
                    'message' => $this->getMessage(),
                ],
            ], $this->getCode());
        }

        return response()->view('errors.' . $this->getCode(), ['exception' => $this], $this->getCode());
    }
}
```

```
class LicenseExpiredException extends BaseException
{
    public const ID = 'EXPIRED';
    protected $code = 401;
    protected $message = 'Given license has expired.'
}
```

Tip given by @Philo01

The tap helper

The `tap` helper is a great way to remove a separate return statement after calling a method on an object. Makes things nice and clean

```
// without tap
$user->update(['name' => 'John Doe']);

return $user;

// with tap()
return tap($user)->update(['name' => 'John Doe']);
```

Tip given by @mattkingshott

Reset all of the remaining time units

You can insert an exclamation into the `DateTime::createFromFormat` method to reset all of the remaining time units

```
// 2021-10-12 21:48:07.0
DateTime::createFromFormat('Y-m-d', '2021-10-12');

// 2021-10-12 00:00:00.0
DateTime::createFromFormat('!Y-m-d', '2021-10-12');

2021-10-12 21:00:00.0
DateTime::createFromFormat('!Y-m-d H', '2021-10-12');
```

Tip given by @SteveTheBauman

Scheduled commands in the console kernel can automatically email their output if something goes wrong

Did you know that any commands you schedule in the console kernel can automatically email their output if something goes wrong

```
$schedule
    ->command(PruneOrganizationsCOmmand::class)
    ->hourly()
    ->emailOutputOnFailure(config('mail.support'));
```

Tip given by @mattkingshott

Be careful when constructing your custom filtered queries using GET parameters

```
if (request()->has('since')) {
    // example.org/?since=
    // fails with illegal operator and value combination
    $query->whereDate('created_at', '<=', request('since'));
}

if (request()->input('name')) {
    // example.org/?name=0
    // fails to apply query filter because evaluates to false
    $query->where('name', request('name'));
}

if (request()->filled('key')) {
    // correct way to check if get parameter has value
}
```

Tip given by @mc0de

Dust out your bloated route file

Dust out your bloated route file and split it up to keep things organized

```
class RouteServiceProvider extends ServiceProvider
{
    public function boot()
    {
        $this->routes(function () {
            Route::prefix('api/v1')
                ->middleware('api')
                ->namespace($this->namespace)
                ->group(base_path('routes/api.php'));

            Route::prefix('webhooks')
                ->namespace($this->namespace)
                ->group(base_path('routes/webhooks.php'));

            Route::middleware('web')
                ->namespace($this->namespace)
                ->group(base_path('routes/web.php'));

            if ($this->app->environment('local')) {
                Route::middleware('web')
                    ->namespace($this->namespace)
                    ->group(base_path('routes/local.php'));
            }
        });
    }
}
```

Tip given by @Philo01

You can send e-mails to a custom log file

In Laravel you can send e-mails to a custom log file.

You can set your environment variables like this:

```
MAIL_MAILER=log
MAIL_LOG_CHANNEL=mail
```

And also configure your log channel:

```
'mail' => [
    'driver' => 'single',
    'path' => storage_path('logs/mails.log'),
    'level' => env('LOG_LEVEL', 'debug'),
],
```

Now you have all your e-mails in /logs/mails.log
It's a good use case to quickly test your mails.

Tip given by @mmartin_joo

Markdown made easy

Laravel provides an interface to convert markdown in HTML out of the box, without the need to install new composer packages.

```
$html = Str::markdown('# Changelogfy')
```

Output:

```
<h1>Changelogfy</h1>
```

Tip given by @paulocastellano

Simplify if on a request with whenFilled() helper

We often write if statements to check if a value is present on a request or not.

You can simplify it with the `whenFilled()` helper.

```
public function store(Request $request)
{
    $request->whenFilled('status', function (string $status)) {
        // Do something amazing with the status here!
    }, function () {
        // This it called when status is not filled
    });
}
```

Tip given by @mmartin_joo

Pass arguments to middleware

You can pass arguments to your middleware for specific routes by appending ':' followed by the value. For example, I'm enforcing different authentication methods based on the route using a single middleware.

```
Route::get('...')->middleware('auth.license');
Route::get('...')->middleware('auth.license:bearer');
Route::get('...')->middleware('auth.license:basic');
```

```
class VerifyLicense
{
    public function  handle(Request $request, Closure $next, $type = null)
    {
        $licenseKey  = match ($type) {
            'basic'  => $request->getPassword(),
            'bearer' => $request->bearerToken(),
            default  => $request->get('key')
        };

        // Verify license and return response based on the authentication type
    }
}
```

Tip given by @Philo01

Get value from session and forget

If you need to grab something from the Laravel session, then forget it immediately, consider using `session()->pull($value)`. It completes both steps for you.

```
// Before
$path = session()->get('before-github-redirect', '/components');

session()->forget('before-github-redirect');

return redirect($path);

// After
return redirect(session()->pull('before-github-redirect', '/components'))
```

Tip given by @jasonlbeggs

$request->date() method

New in this week's Laravel v8.77: `$request->date()` method.
Now you don't need to call Carbon manually, you can do something like: `$post->publish_at = $request->date('publish_at')->addHour()->startOfHour();`
Link to full pr by @DarkGhostHunter

Use through instead of map when using pagination

When you want to map paginated data and return only a subset of the fields, use `through` rather than `map`. The `map` breaks the pagination object and changes it's identity. While, `through` works on the paginated data itself

```
// Don't: Mapping paginated data
$employees = Employee::paginate(10)->map(fn ($employee) => [
    'id' => $employee->id,
    'name' => $employee->name
])

// Do: Mapping paginated data
$employees = Employee::paginate(10)->through(fn ($employee) => [
    'id' => $employee->id,
    'name' => $employee->name
])
```

Tip given by @bhaidar