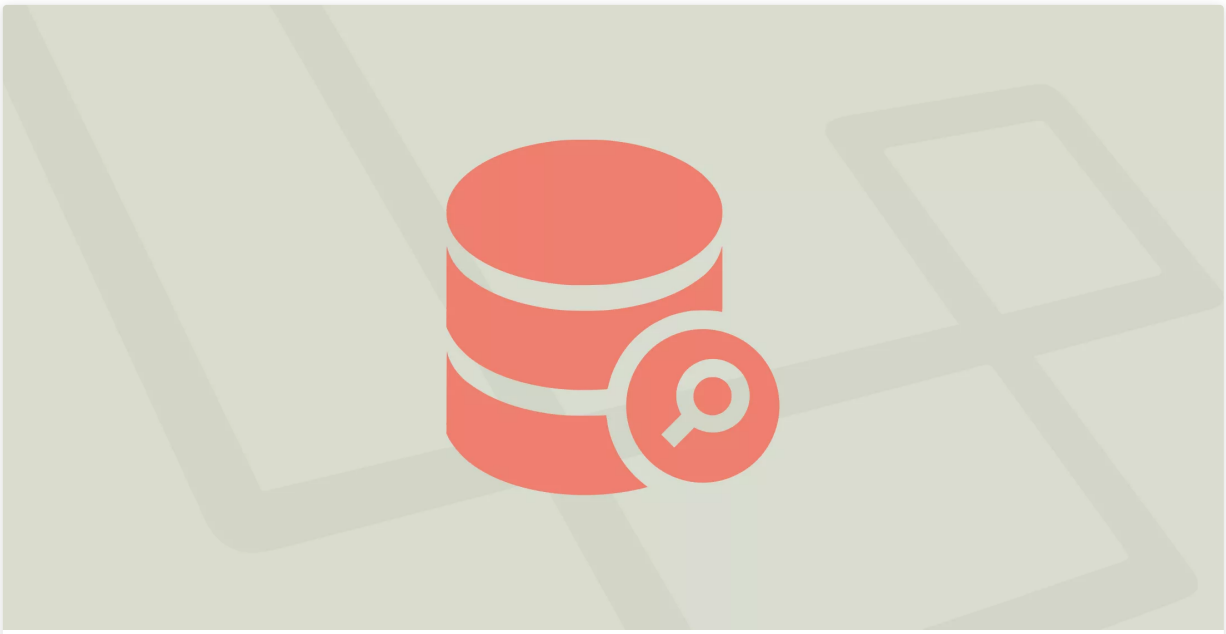


20 Laravel Eloquent Tips and Tricks

OCTOBER 27, 2018 / POVILASKOROP

Eloquent ORM seems like a simple mechanism, but under the hood, there's a lot of semi-hidden functions and less-known ways to achieve more with it. In this article, I will show you a few tricks.

1. Increments and Decrements

Instead of this:

```
$article = Article::find($article_id);
$article->read_count++;
```

You can do this:

```
$article = Article::find($article_id);
$article->increment('read_count');
```

Also these will work:

```
Article::find($article_id)->increment('read_count');
Article::find($article_id)->increment('read_count', 10); // +10
Product::find($produce_id)->decrement('stock'); // -1
```

2. XorY methods

Eloquent has quite a few functions that combine two methods, like “please do X, otherwise do Y”.

Example 1 – `findOrFail()`:

Instead of:

```
$user = User::find($id);
if (!$user) { abort (404); }
```

Do this:

Example 2 – `firstOrCreate()`:

Instead of:

```
$user = User::where('email', $email)->first();
if (!$user) {
  User::create([
    'email' => $email
  ]);
}
```

Do just this:

```
$user = User::firstOrCreate(['email' => $email]);
```

3. Model boot() method

There is a magical place called `boot()` in an Eloquent model where you can override default behavior:

```
class User extends Model
{
    public static function boot()
    {
        parent::boot();
        static::updating(function($model)
        {
            // do some logging
```

Probably one of the most popular examples is setting some field value at the moment of creating the model object. Let's say you want to generate UUID field at that moment.

```
public static function boot()
{
  parent::boot();
  self::creating(function ($model) {
    $model->uuid = (string)Uuid::generate();
  });
}
```

4. Relationship with conditions and ordering

This is a typical way to define relationship:

```
public function users() {
    return $this->hasMany('App\User');
}
```

But did you know that at this point we can already add `where` or `orderBy`?
For example, if you want a specific relationship for some type of users, also ordered by email, you can do this:

```
}
```

5. Model properties: timestamps, appends etc.

There are a few “parameters” of an Eloquent model, in the form of properties of that class. The most popular ones are probably these:

```
class User extends Model {
    protected $table = 'users';
    protected $fillable = ['email', 'password']; // which fields
    protected $dates = ['created_at', 'deleted_at']; // which f
    protected $appends = ['field1', 'field2']; // additional va
}
```

But wait, there’s more:

```
protected $primaryKey = 'uuid'; // it doesn't have to be "id"
public $incrementing = false; // and it doesn't even have to be
protected $perPage = 25; // Yes, you can override pagination cou
const CREATED_AT = 'created_at';
const UPDATED_AT = 'updated_at'; // Yes, even those names can be
public $timestamps = false; // or even not used at all
```

And there’s even more, I’ve listed the most interesting ones, for more please check out the code of default abstract Model class and check out all the traits used.

6. Find multiple entries

```
$user = User::find(1);
```

I'm quite surprised how few people know about that it can accept multiple IDs as an array:

```
$users = User::find([1,2,3]);
```

7. WhereX

There's an elegant way to turn this:

```
$users = User::where('approved', 1)->get();
```

Into this:

```
$users = User::whereApproved(1)->get();
```

Yes, you can change the name of any field and append it as a suffix to "where" and it will work by magic.

Also, there are some pre-defined methods in Eloquent, related to date/time:

```
User::whereMonth('created_at', date('m'));
User::whereYear('created_at', date('Y'));
```

8. Order by relationship

A little more complicated “trick”. What if you have forum topics but want to order them by latest **post**? Pretty common requirement in forums with last updated topics on the top, right?

First, describe a separate relationship for the **latest post** on the topic:

```
public function latestPost()
{
    return $this->hasOne(\App\Post::class)->latest();
}
```

And then, in our controller, we can do this “magic”:

```
$users = Topic::with('latestPost')->get()->sortByDesc('latestPos
```

9. Eloquent::when() – no more if-else’s

Many of us write conditional queries with “if-else”, something like this:

```
}
if (request('filter_by') == 'date') {
    $query->orderBy('created_at', request('ordering_rule', 'desc
}
```

But there's a better way – to use `when()`:

```
$query = Author::query();
$query->when(request('filter_by') == 'likes', function ($q) {
    return $q->where('likes', '>', request('likes_amount', 0));
});
$query->when(request('filter_by') == 'date', function ($q) {
    return $q->orderBy('created_at', request('ordering_rule', 'c
});
```

It may not feel shorter or more elegant, but the most powerful is passing of the parameters:

```
$query = User::query();
$query->when(request('role', false), function ($q, $role) {
    return $q->where('role_id', $role);
});
$authors = $query->get();
```

10. BelongsTo Default Models

Let's say you have Post belonging to Author and then Blade code:

But what if the author is deleted, or isn't set for some reason? You will get an error, something like "property of non-object".

Of course, you can prevent it like this:

```
{{ $post->author->name ?? '' }}
```

But you can do it on Eloquent relationship level:

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

In this example, the `author()` relation will return an empty `App\Author` model if no author is attached to the post.

Furthermore, we can assign default property values to that default model.

```
public function author()
{
    return $this->belongsTo('App\Author')->withDefault([
        'name' => 'Guest Author'
    ]);
}
```

Imagine you have this:

```
function getFullNameAttribute()
{
  return $this->attributes['first_name'] . ' ' . $this->attribut
}
```

Now, you want to order by that `full_name`? This won't work:

```
$clients = Client::orderBy('full_name')->get(); // doesn't work
```

The solution is quite simple. We need to order the results **after** we get them.

```
$clients = Client::get()->sortBy('full_name'); // works!
```

Notice that the function name is different – it's not **orderBy**, it's **sortBy**.

12. Default ordering in global scope

What if you want to have `User::all()` always be ordered by `name` field? You can assign a global scope. Let's go back to the `boot()` method, which we mentioned already above.

```
    parent::boot();

    // Order by name ASC
    static::addGlobalScope('order', function (Builder $builder)
        $builder->orderBy('name', 'asc');
    });
}
```

Read more about Query Scopes here.

13. Raw query methods

Sometimes we need to add raw queries to our Eloquent statements. Luckily, there are functions for that.

```
// whereRaw
$orders = DB::table('orders')
    ->whereRaw('price > IF(state = "TX", ?, 100)', [200])
    ->get();

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

// orderByRaw
User::where('created_at', '>', '2016-01-01')
  ->orderByRaw('(updated_at - created_at) desc')
  ->get();
```

14. Replicate: make a copy of a row

Short one. Without deep explanations, here's the best way to make a copy of database entry:

```
$newTask->save();
```

15. Chunk() method for big tables

Not exactly Eloquent related, it's more about Collection, but still powerful – to process bigger datasets, you can chunk them into pieces.

Instead of:

```
$users = User::all();
foreach ($users as $user) {
    // ...
```

You can do:

```
User::chunk(100, function ($users) {
    foreach ($users as $user) {
        // ...
    }
});
```

16. Create additional things when creating a model

We all know this Artisan command:

```
php artisan make:model Company
```

related files to the model?

```
php artisan make:model Company -mcr
```

- -m will create a **migration** file
- -c will create a **controller**
- -r will indicate that controller should be **resourceful**

17. Override updated_at when saving

Did you know that `->save()` method can accept parameters? As a result, we can tell it to "ignore" `updated_at` default functionality to be filled with current timestamp. See this:

```
$product = Product::find($id);
$product->updated_at = '2019-01-01 10:00:00';
$product->save(['timestamps' => false]);
```

Here we're overriding default `updated_at` with our pre-defined one.

18. What is the result of an update()?

Have you ever wondered what this code actually returns?

```
$result = $products->whereNull('category_id')->update(['category
```

I mean, the update is performed in the database, but what would that `$result` contain?

The answer is **affected rows**. So if you need to check how many rows were affected, you don't need to call anything else – `update()` method will return this number for you.

19. Transform brackets into an Eloquent query

What if you have and-or mix in your SQL query, like this:

```
... WHERE (gender = 'Male' and age >= 18) or (gender = 'Female'
```

How to translate it into Eloquent? This is the **wrong** way:

```
$q->where('gender', 'Male');
$q->orWhere('age', '>=', 18);
$q->where('gender', 'Female');
$q->orWhere('age', '>=', 65);
```

The order will be incorrect. The right way is a little more complicated, using closure functions as sub-queries:

```
$q->where(function ($query) {
    $query->where('gender', 'Male')
        ->where('age', '>=', 18);
})->orWhere(function($query) {
    $query->where('gender', 'Female')
```

20. orWhere with multiple parameters

Finally, you can pass an array of parameters to `orWhere()`. "Usual" way:

```
$q->where('a', 1);
$q->orWhere('b', 2);
$q->orWhere('c', 3);
```

You can do it like this:

```
$q->where('a', 1);
$q->orWhere(['b' => 2, 'c' => 3]);
```

If you enjoyed these Eloquent tips, check out my online course Eloquent: Expert Level and learn about creating relationships, querying data effectively and exploring Eloquent features that you may not know about.

Filed in: Laravel Tutorials / Eloquent

Laravel Jobs

Senior Backend Laravel Engineer

Remote
Kirschbaum Development Group

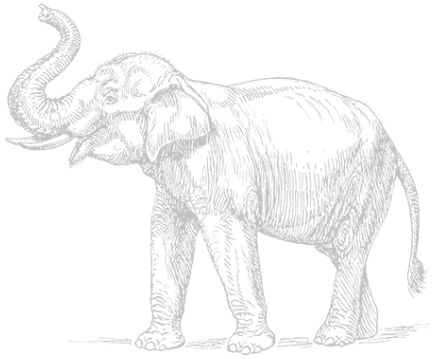
Senior Programmer **Immediate Opening (send resume to daniel@sbgcorp.com)**
Remote
SBG Corp

Full Stack Laravel Developer
London, UK
Keystone Law

Senior Backend developer
Vienna
Identum - Agency for Brand Charism

Leverage Eloquent To Prepare Your URLs
It's not uncommon to have tens, if not hundreds of views in a Laravel application. Something that soon gets out…

Composer v1.6.4 is Released With a Security Fix
The Composer team released v1.6.4 and it includes a security fix so all users are encouraged to upgrade and it also i…



DESIGN & FRONT-END CODE BY