20 Laravel Eloquent порад і трюків

15.12.2019 Cinex Веб-розробка 0

ПОШУК

ПОШУК ...

ПОЗНАЧКИ

APACHE AUTORUN BASH BITCOIN BLOCKCHAIN COMPOSER CRYPTOCURRENCY CSS DATABASE DJANGO DOCKER ELOQUENT ENGLISH ENUM GAMES GIMP GIT HTML JAVA JAVASCRIPT JQUERY LARAVEL MERCURIAL PARSING PHOTOSHOP PHP PHPSTORM

Eloquent ORM здається простим механізмом, але під капотом приховано безліч прихованих функцій і маловідомих способів добитися більшого з його допомогою. У цій статті я покажу вам декілька хитрощів.

1. Збільшення і зменшення

Замість цього:

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

Ти можеш зробити так:

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

Це також буде працювати:

```php
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 методи

Eloquent має кілька функцій, які об'єднують два методи, наприклад «будь ласка, зробіть X, в іншому випадку зробіть Y».

Приклад 1 - findOrFail ():

Замість цього:

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

PLAYONLINUX PLUGINS PROBLEM SOLVING PYTHON REGEXP SEO SOFTWARE ENGINEERING SPRING SQL TELEGRAM TIPS AND TRICKS TIPS AND TRICKS UNIT TESTING WORDPRESS ВІРШІ ШПАРГАЛКИ

ОСТАННІ НОТАТКИ

20 Laravel Eloquent порад і трюків
15.12.2019

Як зробити скріншот сайту по URL на PHP
04.08.2019

Docker Cheat Sheet
05.07.2019

Гарячі клавіші Ubuntu Linux
01.07.2019

Git happens! 6 типових помилок Git і як їх виправити
22.06.2019

НЕДАВНІ КОМЕНТАРІ

це:

```php
$user = User::findOrFail($id);
```

Приклад 2 – firstOrCreate():

Замість цього

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

Можете зробити так

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

3. Model boot() метод

У моделі Eloquent є чарівне місце, зване boot(), де ви можете перевизначити поведінку за замовчуванням:

```php
class User extends Model
{
    public static function boot()
    {
        parent::boot();
        static::updating(function($model)
        {
            // do some logging
            // override some property like $model->something = transform($something);
        });
    }
```

```
12 }
```

Ймовірно, одним з найпопулярніших прикладів є установка деякого значення поля в момент створення модельного об'єкта. Припустимо, ви хочете створити поле UUID в цей момент.

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

4. Зв'язок з умовами і порядком

Це типовий спосіб визначення відносин:

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

Але чи знаєте ви, що в цей момент ми вже можемо додати умову where або orderBy? Наприклад, якщо ви хочете встановити певні відносини для користувачів певного типу, також підтверджених по електронній пошті, ви можете зробити це:

```php
public function approvedUsers() {
    return $this->hasMany('App\User')->where('approved', 1)->orderBy('email');
}
```

5. Властивості моделі: тимчасові мітки, додавання і т. Д.

Є кілька «параметрів» моделі Eloquent в формі властивостей цього класу. Найпопулярніші з них:

```php
class User extends Model {
```

```php
    protected $table = 'users';
    protected $fillable = ['email', 'password']; // які поля можуть бути заповнені User::
    protected $dates = ['created_at', 'deleted_at']; // які поля будуть Carbon-ized
    protected $appends = ['field1', 'field2']; // додаткові значення повертаються в JSON
}
```

Но подождите, это еще не все:

```php
protected $primaryKey = 'uuid'; // це не повинно бути "id"
public $incrementing = false; // і це навіть не повинно бути auto-incrementing!
protected $perPage = 25; // Так, ви можете змінити лічильник нумерації сторінок PER MODEL
const CREATED_AT = 'created_at';
const UPDATED_AT = 'updated_at'; // Так, навіть ці імена можуть бути перевизначені
public $timestamps = false; // або навіть не використовується взагалі
```

І ще більше, я перерахував найцікавіші з них, для більш докладної інформації ознайомтеся з кодом **абстрактного класу Model** за замовчуванням і перевірте всі використовувані можливості.

6. Знайти кілька записів

Всі знають метод find(), вірно?

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

Я дуже здивований, як мало хто знає про те, що він може приймати кілька ідентифікаторів у вигляді масиву:

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

7. WhereX

Є елегантний спосіб перетворити це:

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

в це:

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

Так, ви можете змінити ім'я будь-якого поля і додати його в якості суфікса до «where», і це буде працювати як магія.

Кроме того, в Eloquent є кілька визначених методів, пов'язаних з датою і часом:

```php
User::whereDate('created_at', date('Y-m-d'));
User::whereDay('created_at', date('d'));
User::whereMonth('created_at', date('m'));
User::whereYear('created_at', date('Y'));
```

8. Order by relationship

Трохи складніший «трюк». Що якщо у вас є теми на форумі, але ви хочете відсортувати їх за останнім постом? Досить поширене вимога на форумах з останніми оновленими темами вгорі, вірно?

Спочатку опишіть окреме ставлення до останнього допису по темі:

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

І тоді, в нашому контролері, ми можемо зробити це «диво»:

```php
users = Topic::with('latestPost')->get()->sortByDesc('latestPost.created_at');
```

9. Eloquent::when () - більше ніяких if-else

Багато з нас пишуть умовні запити за допомогою if-else, щось на зразок цього:

```php
if (request('filter_by') == 'likes') {
    $query->where('likes', '>', request('likes_amount', 0));
}
if (request('filter_by') == 'date') {
    $query->orderBy('created_at', request('ordering_rule', 'desc'));
}
```

Але є кращий спосіб - використовувати when():

```php
$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', 'desc'));
});
```

Можливо, він не буде виглядати коротше або елегантніше, але найпотужнішим є передача параметрів:

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

10. BelongsTo Default Models

Припустимо, у вас є пост, що належить автору, а потім код Blade:

```php
{{ $post->author->name }}
```

Але що, якщо автор видалений або не встановлений за будь-якої причини? Ви отримаєте помилку, щось на зразок «property of non-object».

Звичайно, ви можете запобігти цьому наступним чином:

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

Але ви можете зробити це на рівні відносин Eloquent:

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

У цьому прикладі ставлення author() поверне порожню модель App\Author, якщо до записі не прикріплений жоден автор.

Крім того, ми можемо присвоїти значення властивостей за замовчуванням для цієї моделі.

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

11. Order by Mutator

Уявіть, що у вас є це:

```php
function getFullNameAttribute()
{
  return $this->attributes['first_name'] . ' ' . $this->attributes['last_name'];
```

```
}
```

Тепер ви хочете відсортувати за цим повним ім'ям? Це не спрацює:

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

Рішення досить просте. Нам потрібно впорядкувати результати після того, як ми їх отримаємо.

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

Зверніть увагу, що ім'я функції відрізняється - це не **orderBy**, а **sortBy**.

12. Порядок за умовчанням в глобальному контексті

Що якщо ви хочете, щоб User::all() завжди впорядковує по полю імені? Ви можете призначити глобальну область. Давайте повернемося до методу boot(), про який ми вже згадували вище.

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

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

Дізнайтеся більше про Query Scopes тут.

13. Raw query methods

Іноді нам потрібно додати необроблені запити до наших заяв Eloquent. На щастя, для цього є функції.

```php
// 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. Копіювати: зробити копію рядка

Коротко. Без глибоких пояснень, ось кращий спосіб зробити копію запису бази даних:

```php
$task = Tasks::find(1);
$newTask = $task->replicate();
$newTask->save();
```

15. Chunk() метод для великих таблиць

Не зовсім пов'язаний з Eloquent, він більше стосується Collection, але все ж потужний - для обробки великих наборів даних ви можете розбити їх на частини.

Вместо того:

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

Ти можеш зробити:

```php
User::chunk(100, function ($users) {
```

```
    foreach ($users as $user) {
        // ...
    }
});
```

16. Створіть додаткові речі при створенні моделі

Ми всі знаємо цю команду Artisan:

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

Але чи знаєте ви, що є три корисних прапора для генерації пов'язаних файлів з моделлю?

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

-m створить файл міграції
-c створить контроллер
-r буде вказувати, що контролер повинен бути resourceful

17. Перевизначите updated_at при збереженні

Чи знаєте ви, що метод ->save() може приймати параметри? В результаті ми можемо сказати йому «ігнорувати» оновлену функціональність за замовчуванням updated_at для заповнення поточної відміткою часу. Подивимося на це:

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

Тут ми перевизначаємо default_at за замовчуванням на наше заздалегідь певне значення

18. Який результат update()?.

Чи замислювалися ви, що цей код насправді повертає?

```php
$result = $products->whereNull('category_id')->update(['category_id' => 2]);
```

Я маю на увазі, що оновлення виконується в базі даних, але що буде містити цей результат?

Відповідь: скільки було затронуто рядків. Тому, якщо вам потрібно перевірити, скільки рядків було затронуто, вам не потрібно більше нічого викликати - метод update() поверне вам це число.

19. Перетворіть дужки в Eloquent query

Що робити, якщо у вас є і / або змішати в вашому запиті SQL, як це:

```mysql
... WHERE (gender = 'Male' and age >= 18) or (gender = 'Female' and age >= 65)
```

Як перевести це на Eloquent? Це неправильний шлях:

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

Порядок буде невірним. Правильний шлях трохи складніше, використовуючи функції замикання в якості підзапитів:

```php
$q->where(function ($query) {
    $query->where('gender', 'Male')
        ->where('age', '>=', 18);
})->orWhere(function($query) {
    $query->where('gender', 'Female')
        ->where('age', '>=', 65);
})
```

20. orWhere з декількома параметрами

Нарешті, ви можете передати масив параметрів в orWhere ().
«Звичайний» спосіб:

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

Ви можете зробити це так:

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

Джерело

Переклад: Cinex

ELOQUENT LARAVEL TIPS AND TRICKS

« ПОПЕРЕДНІЙ
Як зробити скріншот сайту по URL на PHP

ЗАЛИШТЕ ПЕРШИЙ КОМЕНТАР

Залишити коментар

Вашу адресу електронної пошти не буде опубліковано

Коментувати

Ім'я*

Email *

Збережіть моє ім'я, електронну пошту у цьому веб-переглядачі наступного разу, коли я коментуватиму.

ОПУБЛІКУВАТИ КОМЕНТАР