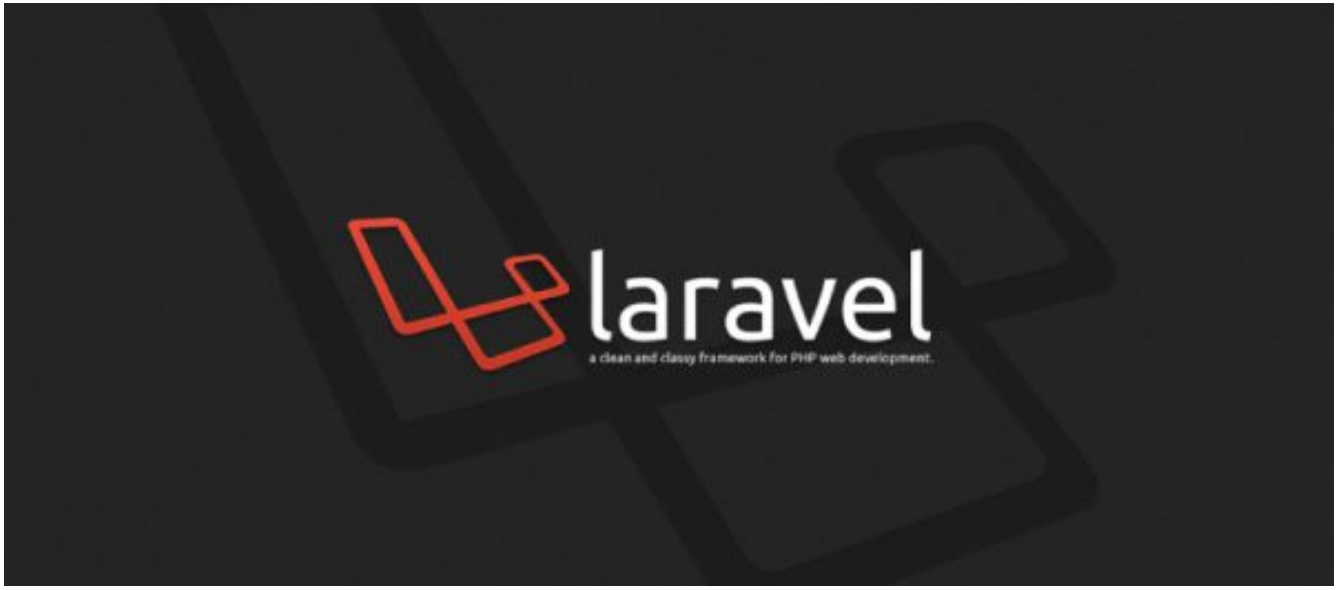
20 хитрощів в Laravel Eloquent про які ви не знали

16.04.2019 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 існує багато хитрих функцій і способів досягнення різних цілей. У цій статті я покажу вам кілька трюків.

1. Інкремент і декременти

Замість цього:

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

Ви можете робити так:

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

І навіть ось так:

```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

У Eloquent існує кілька методів які можна об'єднує в один, щось на зразок "Зроби X, і в разі невдачі зроби Y".

Пример 1: findOrFail():

Замість цього:

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

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

Ви можете робити так:

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

Пример 2: findOrCreate():

Замість ось такого:

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

Краще робити ось так:

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

3. Метод boot()

У методі boot() ви можете перевизначити поведінку Eloquent моделі:

```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);
```

```
    });
  }
}
```

Самим розповсюдженим прикладом використання методу boot() - встановлення значення за умовчанням для будь-якого поля. Наприклад, згенеруємо UUID в момент створення моделі:

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

4. Relationship з умовами та сортуванням

Ось так зазвичай визначають відносини:

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

Але чи знали ви, що вже на даному етапі можна додати **orderBy**? Наприклад, якщо ми хочемо додати relation тільки для певного типу користувачів, з сортуванням по email, то ми можемо зробити так:

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

5. Властивості моделі: timestmap, appends, і т.д.

У Eloquent моделі є кілька властивостей, про які багато хто не знає. Найпопулярніші:

```php
class User extends Model {
  protected $table = 'users';
  protected $fillable = ['email', 'password']; // which fields can be filled with User::c
  protected $dates = ['created_at', 'deleted_at']; // which fields will be Carbon-ized
  protected $appends = ['field1', 'field2']; // additional values returned in JSON
}
```

Але існують і інші, такі як:

```php
protected $primaryKey = 'uuid'; // Если у вас нету поля 'id'
public $incrementing = false; // Указывает что primary key не имеет свойства auto increme
protected $perPage = 25; // А тут мы переопределяем кол-во результатов на странице при па
const CREATED_AT = 'created_at'; const UPDATED_AT = 'updated_at'; // А этими константами
public $timestamps = false; // или можно указать что мы вообще не используем поля created
```

Насправді існує набагато більше подібних властивостей, я перерахував лише кілька. Щоб дізнатися про інші властивості **відкрийте код** і перегляньте все трейти які використовуються.

6. Пошук декількох записів

Всі знають про метод find(), вірно?

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

Але я був здивований, що дуже мало хто знає що цей метод приймає так само масив з декількома ключами:

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

7. WhereX

У Eloquent існує елегантний спосіб перетворити це:

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

В це:

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

Так, правильно, ми можемо взяти ім'я будь-якого поля в нашій моделі і додати його в якості суфікса до методу *where*.

Так само існує кілька подібних методів для роботи з датами, які Eloquent містить спочатку:

```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. Сортування по relation

Більш складний трюк. Що якщо у нас є форумні теми і ми хоті впорядкувати їх **за датою останнього поста**? Досить таки буденна задача, вірно?

Для початку ми пропишемо окрему зв'язок latest post в моделі теми:

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

І потім, в нашому контролері, ми можемо творити магію:

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

9. Eloquent::when() - избавляется от if-else

Багато з нас пишуть подібні умовні конструкції:

```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 більш витончений спосіб це зробити:

```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'));
});
```

10. BelongsTo і Модель за замовчуванням

Уявімо, що у нас є модель Post, яка belongsTo до моделі Author, і ми виводимо автора в шаблоні:

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

Але що трапиться якщо модель автора видалена або null з будь-якої причини? Буде помилка! Звичайно ми може зробити перевірку:

```
//
```

Але тоді нам доведеться робити такі перевірки в кожному місці де ми хочемо отримати ім'я автора. Є більш витончений спосіб зробити це за допомогою Eloquent:

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

В даному прикладі relation author() поверне порожню модель Author, якщо в Post немає зв'язку з реальною моделлю. Ми можемо навіть вказати властивості за замовчуванням для цієї порожній моделі:

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

11. Order by Mutator

Уявімо що у нас є такий код:

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

Ви хочете зробити сортування по full_name? Такий підхід не спрацює:

```php
$clients = Client::orderBy('full_name')->get(); // так не працює
```

Рішення досить просте. Ми повинні сортувати результати **ПІСЛЯ** того як ми їх отримали:

```php
$clients = Client::get()->sortBy('full_name'); // а ось так працює!
```

Зверніть увагу що ми використовуємо не **orderBy**, а **sortBy**, функцію з *Collection*.

12. Сортування за замовчуванням для глобального scope

Що якщо ми хочемо, щоб запит *User::all()* завжди був впорядкований по полю name? Ми можемо призначити глобальний scope для визначення такої поведінки. Давайте повернемося до методу *boot()*, про який ми говорили раніше, і використовуємо його:

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

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

Детальніше можна почитати в документації.

13. Raw запити

Іноді нам потрібно здійснювати "сирі" (raw) запити до бази даних. На щастя Eloquent підтримує і їх:

```php
// whereRaw
```

```php
$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: створити копію запису

Дуже коротко: ось так можна створити копію запису в базі даних:

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

15. Метод Chunk для великих таблиць

Це більше відноситься до колекцій, а не до Eloquent, але все одно. Якщо у вас є велика таблиця (з тисячами записів) і ви не можете отримати їх всіх за один запит, то можна використовувати метод chunk, який буде діставати записи по "чуть-чуть":

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

16. Створюємо додаткові речі при створенні моделі

Всі ми знаємо про команду **php artisan make:model Company**. Але чи знали ви, що ви відразу можете згенерувати: міграцію, контролер, і навіть вказати що контролер повинен бути REST? Використовуючи додаткові прапори ми можемо зробити це однією командою:

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

Де прапори означають:

- m - створити файл **міграції**
- c - створити **контролер**
- r - контролер повинен бути **REST**

17. Не оновлювати update_at при збереженні

Чи знали ви, що метод *->save()* приймає додаткові параметри? Наприклад, ми можемо вказати що не потрібно оновлювати timestamps, якщо нам це не потрібно:

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

Такий код не буде оновлювати update_at при виклику *save()*.

18. Дізнатися результат update()

Чи цікавилися ви коли-небудь питанням що повертає такий код:

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

Я маю на увазі не що конкретно робить метод *update()*, а що в результаті буде в змінній *$result*?

Відповідь: **к-ть порушених рядків!** Так що якщо ви захочете дізнатися скільки рядків оновив ваш виклик *update()* - вам не потрібно нічого робити! Ви можете просто подивитися на значення яке він повернув.

19. Перетворення скобок з SQL в Eloquent запит

Що якщо у вас є подібний SQL запит:

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

Як правильно скласти Eloquent запит для нього? Ось так **неправильно**:

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

У вас вийде неправильний SQL запит. Правильний спосіб трохи складніший, і полягає в використанні замикань:

```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

На завершення: ви можете передавати масив параметрів в *orWhere()* метод. Зазвичай роблять так:

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

Ви ж можете бути більш крутими, і робити ось так:

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

Я впевнений що існує ще багато трюків, про які ми не знаємо. Можливо хтось поділиться ними в коментарях.

Оригінал на: laravel-news.com

ELOQUENT LARAVEL TIPS AND TRICKS TIPS AND TRICKS

« **ПОПЕРЕДНІЙ**
25 порад і хитрощів в Laravel

ДАЛІ »
Шпаргалка по Git, в якій представлені основні команди