Course - Laravel Framework

Collections Dữ Liệu

Tất cả các collection cũng đóng vai trò là trình vòng lặp (iterator), cho phép bạn lặp lại chúng như thể chúng là các mảng PHP đơn giản.

Tags: laravel, collection, collections du lieu

Giới thiệu

Tất cả các phương thức Eloquent trả về nhiều hơn một model kết quả mà sẽ trả về các đối tượng của class `Illuminate\Database\Eloquent\Collection`, bao gồm các kết quả được truy xuất thông qua phương thức `get` hoặc được truy cập thông qua một mối quan hệ. Đối tượng Eloquent collection mở rộng collection cơ sở của Laravel, do đó, nó thừa hưởng một cách tự nhiên hàng chục phương thức được sử dụng để làm việc trôi chảy với mảng cơ bản của các model Eloquent. Hãy nhớ xem lại tài liệu collection Laravel để tìm hiểu tất cả về các phương thức hữu ích này!

Tất cả các collection cũng đóng vai trò là *trình vòng lặp* (iterator), cho phép bạn lặp lại chúng như thể chúng là các mảng PHP đơn giản:

```
use App\Models\User;

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

foreach ($users as $user) {
  echo $user->name;
}
```

Tuy nhiên, như đã đề cập trước đây, collection mạnh hơn nhiều so với mảng và công khai nhiều lệnh map (bố trí) / thu nhỏ (reduce) có thể được xâu chuỗi thêm bằng cách sử dụng giao diện trực quan. Ví dụ: chúng ta có thể xóa tất cả các model không hoạt động và sau đó thu thập tên cho từng người dùng còn lại:

```
$names = User::all()->reject(function ($user) {
  return $user->active === false;
})->map(function ($user) {
  return $user->name;
});
```

Sự chuyển đổi Eloquent Collection

Trong khi hầu hết các phương thức Eloquent collection mà sẽ trả về một đối tượng mới của Eloquent collection, thì các phương thức như `collapse`, `flatten`, `flip`, `keys`, `pluck` và `zip` sẽ trả về một đối tượng collection cơ sở. Tương tự như vậy, nếu một lệnh bố trí chẳng

hạn như `map` trả về một collection không chứa bất kỳ Eloquent model nào, thì nó sẽ được chuyển đổi thành một đối tượng collection cơ sở.

Các phương thức có sẵn

Tất cả các collection Eloquent đều mở rộng đối tượng collection Laravel cơ sở; do đó, chúng kế thừa tất cả các phương thức linh hoạt mà được cung cấp bởi class collection cơ sở.

Ngoài ra, class `Illuminate\Database\Eloquent\Collection` cung cấp một tập hợp các phương thức để hỗ trợ việc quản lý các collection chứa các model của bạn. Hầu hết các phương thức đều trả về các đối tượng `Illuminate\Database\Eloquent\Collection`; tuy nhiên, một số phương thức, như `modelKeys`, trả về một đối tượng `Illuminate\Support\Collection`.

`contains($key, $operator = null, $value = null)`

Phương thức `contains` có thể được sử dụng để xác định xem một đối tượng model nào đó có được chứa trong collection hay không. Phương thức này chấp nhận một khóa chính hoặc một đối tượng model nào đó:

```
$users->contains(1);

$users->contains(User::find(1));
```

`diff($items)`

Phương thức `diff` sẽ trả về tất cả các model không có trong collection đã cho:

```
use App\Models\User;

$users = $users->diff(User::whereIn('id', [1, 2, 3])->get());
```

`except($keys)`

Phương thức `except` sẽ trả về tất cả các model không có khóa chính (*primary key*) đã cho:

```
$users = $users->except([1, 2, 3]);
```

`find($key)`

Phương thức `find` sẽ trả về model có khóa chính (*primary key*) khớp với khóa đã cho. Nếu $key là một đối tượng model, find sẽ cố gắng trả về một model khớp với khóa chính. Nếu $key là một mảng các khóa, thì hàm find sẽ trả về tất cả các model có khóa chính trong mảng đã cho:

```
$users = User::all();

$user = $users->find(1);
```

`fresh($with = [])`

Phương thức `fresh` lấy ra một đối tượng hoàn toàn mới của mỗi model trong collection từ cơ sở dữ liệu. Ngoài ra, bất kỳ mối quan hệ cụ thể nào cũng sẽ được tải nhanh (*eager loading*):

```
$users = $users->fresh();

$users = $users->fresh('comments');
```

`intersect($items)`

Phương thức `intersect` sẽ trả về tất cả các model cũng có trong collection đã cho:

```
use App\Models\User;

$users = $users->intersect(User::whereIn('id', [1, 2, 3])->get());
```

`load($relations)`

Phương thức `load` sẽ tải eager các mối quan hệ đã cho cho tất cả các model trong collection:

```
$users->load(['comments', 'posts']);

$users->load('comments.author');
```

`loadMissing($relations)`

Phương thức `loadMissing` mong muốn tải các mối quan hệ đã cho cho tất cả các model trong collection nếu các mối quan hệ chưa được tải:

```
$users->loadMissing(['comments', 'posts']);

$users->loadMissing('comments.author');
```

`modelKeys()`

Phương thức `modelKeys` trả về các khóa chính (*primary key*) cho tất cả các model trong collection:

```
$users->modelKeys();

// [1, 2, 3, 4, 5]
```

`makeVisible($attributes)`

Phương thức `makeVible` giúp hiển thị các thuộc tính attribute thường được "ẩn" trên mỗi model trong collection:

```
$users = $users->makeVisible(['address', 'phone_number']);
```

`makeHidden($attributes)`

Phương thức `makeHidden` ẩn các thuộc tính attribute thường "hiển thị" trên mỗi model trong collection:

```
$users = $users->makeHidden(['address', 'phone_number']);
```

`only($keys)`

Phương thức `only` sẽ trả về tất cả các model có khóa chính (*primary key*) đã cho:

```php
$users = $users->only([1, 2, 3]);
```

`toQuery()`

Phương thức `toQuery` sẽ trả về một đối tượng của Eloquent query builder mà chứa ràng buộc `whereIn` trên các khóa chính (*primary key*) của model trong collection:

```php
use App\Models\User;

$users = User::where('status', 'VIP')->get();

$users->toQuery()->update([
  'status' => 'Administrator',
]);
```

`unique($key = null, $strict = false)`

Phương thức `unique` sẽ trả về tất cả các model chỉ có duy nhất trong collection. Mọi model cùng loại có cùng khóa chính (*primary key*) nhưng trong model khác sẽ bị xóa khỏi collection:

```php
$users = $users->unique();
```

Collection tùy chỉnh

Nếu bạn muốn sử dụng đối tượng `Collection` tùy chỉnh khi tương tác với một model nào đó, bạn có thể xác định phương thức `newCollection` trên model của mình:

```php
<?php

namespace App\Models;

use App\Support\UserCollection;
use Illuminate\Database\Eloquent\Model;
```

```php
class User extends Model
{
  /**
   * Create a new Eloquent Collection instance.
   *
   * @param  array  $models
   * @return \Illuminate\Database\Eloquent\Collection
   */
  public function newCollection(array $models = [])
  {
    return new UserCollection($models);
  }
}
```

Khi bạn đã khai báo phương thức `newCollection`, bạn sẽ nhận được một đối tượng của collection tùy chỉnh của mình bất cứ lúc nào. Eloquent thường trả về một đối tượng `Illuminate\Database\Eloquent\Collection`. Nếu bạn muốn sử dụng một collection tùy chỉnh cho mọi model trong ứng dụng của mình, bạn nên xác định phương thức `newCollection` trên một class model cơ sở được mở rộng bởi tất cả các model ứng dụng của bạn.