Bài 6
Eloquent nâng cao

Module: BOOTCAMP WEB-BACKEND DEVELOPMENT WITH LARAVEL 2.1

Mục tiêu

- Định nghĩa được các quan hệ trong ORM
- Truy vấn được dữ liệu từ nhiều bảng
- Truy vấn được dữ liệu với nhiều điều kiện

Các mối quan hệ trong Eloquent

Các loại quan hệ

- Các bảng cơ sở dữ liệu thường liên quan tới nhau
- Eloquent giúp cho việc quản lý và làm việc với các mối quan hệ trở nên dễ dàng hơn
- Các loại relationship khác nhau:
  - One to One
  - One to Many
  - Many to Many
  - Has Many Through

Định nghĩa relationships

- Eloquent Relationships được định nghĩa là các function trong các lớp Eloquent model
- Giống như các Eloquent model, các relationships cũng hỗ trợ query builder
- Định nghĩa relationship cung cấp các method chaining và query
- Ví dụ:

```
$user->posts()->where('active', 1)->get();
```

One to One: Quan hệ 1-1

- Mối quan hệ 1-1 là mối quan hệ hết sức cơ bản giữa một model với một model khác
- Ví dụ, một User có thể được liên kết với 1 Phone
  - Đặt một method phone trên User model
  - Phương thức phone() trả về kết quả của phương thức hasOne()

```php
class User extends Model
{
    /**
     * Get the phone associated with the user.
     */
    public function phone()
    {
        return $this->hasOne(Phone::class);
    }
}
```

Truy cập dynamic properties

- Có thể truy xuất các bản ghi sử dụng các thuộc tính động (dynamic properties
- Dynamic properties cho phép truy cập vào các relationship functions như thể nó là thuộc tính được định nghĩa trên các model
- Ví dụ:

```
$phone = User::find(1)->phone;
```

Tuỳ biến tên các cột liên kết

- Eloquent ngầm định các foreign key dựa theo tên model
- Ví dụ, Phone sẽ tự động lấy foreign key là user_id
  - Có thể thay đổi cột user_id bằng cách thêm đối số thứ 2 của phương thức hasOne:

```
return $this->hasOne(Phone::class, 'foreign_key');
```

- Eloquent ngầm định foreign key sẽ tương ứng với một giá trị của cột id của model hiện tại
- Ví dụ, cột id của User tương ứng với cột user_id của các bản ghi Phone
  - Có thể sử dụng một cột khác với cột id:

```
return $this->hasOne(Phone::class, 'foreign_key', 'local_key');
```

Định nghĩa Inverse của Relationship

- Có thể truy cập vào Phone model từ User model
- Định nghĩa 1 relationship trên Phone model để cho phép truy cập vào User model
- Phương thức nghịch đảo của hasOne là belongsTo
- Ví dụ:

```php
class Phone extends Model
{
    /**
     * Get the user that owns the phone.
     */
    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

Tuỳ chỉnh tên các cột liên kết

- Có thể tuỳ chỉnh tên các cột liên kết thay cho tên ngầm định
- Ví dụ:

```php
/**
 * Get the user that owns the phone.
 */
public function user()
{
    return $this->belongsTo(User::class, 'foreign_key');
}
```

- và:

```php
public function user()
{
    return $this->belongsTo(User::class, 'foreign_key', 'owner_key');
}
```

One To Many: Quan hệ 1-nhiều

- Được sử dụng để quy định các mối quan hệ khi một model sở hữu nhiều số lượng của model khác
  - Ví dụ, 1 blog post có nhiều comment
- Quan hệ One-to-many được xác định bằng 1 function được đặt ở model
- Ví dụ:

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

  - Foreign_key trên Comment model được ngầm định là post_id

Truy xuất thuộc tính động

- Có thể truy cập lấy danh sách comments bằng cách truy cập thuộc tính comments:

```php
use App\Models\Post;

$comments = Post::find(1)->comments;

foreach ($comments as $comment) {
    //
}
```

- Hoặc thêm các lệnh truy vấn:

```php
$comment = Post::find(1)->comments()
                    ->where('title', 'foo')
                    ->first();
```

Tuỳ chỉnh các cột liên kết

- Có thể ghi đè các foreign key là local key bằng cách thêm các đối số cho phương thức hasMany

Ví dụ:

```php
return $this->hasMany(Comment::class, 'foreign_key');

return $this->hasMany(Comment::class, 'foreign_key', 'local_key');
```

Định nghĩa Inverse của quan hệ

- Định nghĩa một quan hệ để cho phép comments có thể truy xuất 1 post cha của nó
- Sử dụng phương thức belongsTo để xác định các inverse của quan hệ hasMany
- Ví dụ:

```
class Comment extends Model
{
    public function post()
    {
        return $this->belongsTo('App\Post');
    }
}
```

Sử dụng dynamic property

- Ví dụ:

```php
use App\Models\Comment;

$comment = Comment::find(1);

return $comment->post->title;
```

- Eloquent sẽ match một post_id từ Comment model với 1 id của Post model

Tuỳ biến tên cột liên kết

- Eloquent ngầm định tên foreign key bằng cách kiểm tra tên của phương thức relationship và nối với hậu tố _id.
- Có thể tùy chỉnh bằng cách thêm đối số thứ 2 trong phương thức belongsTo
- Ví dụ:

```php
public function post()
{
    return $this->belongsTo(Post::class, 'foreign_key', 'owner_key');
}
```

Many To Many: Quan hệ nhiều-nhiều

- Model A thuộc nhiều model B và ngược lại
- Ví dụ, quan hệ giữa user và role:
  - 1 user có nhiều roles
  - 1 role cũng thuộc về nhiều user
- Cần thiết phải có 3 bảng: users, roles và user_role
- Bảng user_role chứa 2 cột user_id và role_id.

Khai báo quan hệ nhiều-nhiều

- Quan hệ many-to-many được định nghĩa bằng cách gọi phương thức belongsToMany của Eloquent
- Ví dụ:

```php
class User extends Model
{
    /**
     * The roles that belong to the user.
     */
    public function roles()
    {
        return $this->belongsToMany(Role::class);
    }
}
```

Truy cập thuộc tính động

- Có thể truy cập vào roles bằng cách truy cập dynamic property
- Ví dụ:

```php
use App\Models\User;

$user = User::find(1);

foreach ($user->roles as $role) {
    //
}
```

- Có thể thêm các lệnh truy vấn, ví dụ:

```php
$roles = App\User::find(1)->roles()->orderBy('name')->get();
```

Tuỳ chỉnh bảng trung gian

- Eloquent ngầm định sử dụng tên của bảng trung gian bằng cách ghép tên của 2 bảng thành phần
- Thứ tự được ghép tuân theo bảng chữ cái
- Có thể ghi đè quy ước này bằng cách thêm vào 1 đối số thứ 2 trong phương thức belongsToMany
- Ví dụ:

```
return $this->belongsToMany('App\Role', 'role_user');
```

  - Hoặc tuỳ biến tên các cột:

```
return $this->belongsToMany('App\Role', 'role_user', 'user_id', 'role_id');
```

Định nghĩa Inverse của quan hệ

- Sử dụng phương thức belongsToMany để khai báo nghịch đảo của mối quan hệ many-to-many
- Ví dụ:

```php
class Role extends Model
{
    /**
    * The users that belong to the role.
    */
    public function users()
    {
        return $this->belongsToMany('App\User');
    }
}
```

Truy xuất dữ liệu các cột của bảng trung gian

- Có thể truy cập vào bảng trung gian bằng cách sử dụng thuộc tính **pivot**
- Ví dụ:

```
$user = App\User::find(1);
foreach ($user->roles as $role) {
    echo $role->pivot->created_at;
}
```

- Nếu bảng pivot chứa các thuộc tính mở rộng, cần phải xác định chúng tại thời điểm khai báo mối quan hệ, ví dụ:

```
return $this->belongsToMany('App\Role')->withPivot('column1', 'column2');
```

Thêm các trường timestamp

- Sử dụng các phương thức Timestamps để bảng pivot tự động có created_at và updated_at
- Ví dụ:

```
return $this->belongsToMany('App\Role')->withTimestamps();
```

Lọc kết quả trả về dựa trên bảng trung gian

- Sử dung wherePivot. Ví dụ:

```
return $this->belongsToMany('App\Role')->withTimestamps();
```

- Sử dụng wherePivotIn. Ví dụ:

```
return $this->belongsToMany('App\Role')->wherePivot('approved', 1);
```

Quan hệ: Has Many Through

- Cung cấp tiện ích để truy cập vào các mối quan hệ xa thông qua một mối quan hệ trung gian
- Ví dụ:

countries
id - integer
name - string

users
id - integer
country_id - integer
name - string

posts
id - integer
user_id - integer
title - string

- Cần lấy các *posts* của một country thông qua bảng trung gian *users*

Has Many Through: Ví dụ

```php
class Country extends Model
{
    public function posts()
    {
        return $this->hasManyThrough('App\Post', 'App\User');
    }
}
```

Truy vấn dữ liệu từ nhiều bảng

Eloquent tự động thực hiện join

- Thao tác join giữa các bảng liên quan được thực hiện tự động
- Ví dụ: Lấy danh sách các comment của một post

```
$comments = App\Post::find(1)->comments;

foreach ($comments as $comment) {
        //TODO:
 }
```

- Hoặc tìm comment đầu tiên của một post với *title* là *foo*:

```
$comment = App\Post::find(1)->comments()->where('title', 'foo')->first();
```

Các hàm khi truy vấn

Danh sách các hàm thông dụng

Tên hàm	Ý nghĩa
all	Trả về tất cả các thực thể
first	Trả về thực thể đầu tiên
last	Trả về thực thể cuối cùng
where	Lọc theo điều kiện một giá trị cho trước
whereIn	Lọc theo điều kiện một giá trị nằm trong một danh sách cho trước
whereNotIn	Lọc theo điều kiện một giá trị không nằm trong một danh sách cho trước
whereBetween	Lọc theo điều kiện một giá trị nằm trong khoảng cho trước
has	Xác định liệu một danh sách có chứa một thực thể cho trước
count	Trả về số lượng các thực thể

Truy vấn dựa trên sự tồn tại của thực thể con

- Có thể lọc kết quả của truy vấn dựa trên sự tồn tại của thực thể con
- Ví dụ: Lấy danh sách những post trong đó có comment chứa tiêu đề là *foo*

```
$posts = App\Post::whereHas('comments', function ($query) {

    $query->where('content', 'like', 'foo%’);

})->get();
```

Đếm số lượng của thực thể con

- Sử dung hàm withCount() để lấy về số lượng của thực thể con
- Ví dụ: Lấy về danh sách các post kèm theo số lượng comment của từng post

```
$posts = App\Post::withCount('comments')->get();

foreach ($posts as $post) {

    echo $post->comments_count;

}
```

Tóm tắt bài học

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***Validation***