**----------------------------------------------------------------------------------------------------------------------------------Tải về làm chứ không làm trực tiếp được đâu :v**

|  |  |  |
| --- | --- | --- |
|  | **ORM và Eloquent nâng cao** | |
|  | Các mối quan hệ trong Eloquent | **1-1;1-n;n-n;n-1** |
|  | Eloquent Relationships là gì ? | Mối quan hệ Eloquent được định nghĩa là các phương thức trên các lớp mô hình Eloquent của bạn |
|  | Cách triển khai One - One | Để xác định mối quan hệ này, chúng ta sẽ đặt một phonephương thức trên Usermô hình. Các phonephương pháp nên gọi hasOnephương thức và trả về kết quả của nó  **<?php**  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  class User extends Model  {  /\*\*  \* Get the phone associated with the user.  \*/  public function phone()  {  return $this->hasOne(Phone::class);  }  }  Eloquent xác định khóa ngoại của mối quan hệ dựa trên tên mô hình mẹ. Trong trường hợp này, Phonemô hình tự động được giả định là có user\_idkhóa ngoại. Nếu bạn muốn ghi đè quy ước này, bạn có thể chuyển đối số thứ hai cho hasOnephương thức  return $this->hasOne(Phone::class, 'foreign\_key');  Ngoài ra, Eloquent giả định rằng khóa ngoại phải có giá trị khớp với cột khóa chính của khóa cha. Nói cách khác, Eloquent sẽ tìm kiếm giá trị của idcột của người dùng trong user\_idcột của Phonebản ghi. Nếu bạn muốn mối quan hệ sử dụng giá trị khóa chính khác với idhoặc thuộc tính mô hình của $primaryKeybạn, bạn có thể chuyển đối số thứ ba cho hasOnephương thức  return $this->hasOne(Phone::class, 'foreign\_key', 'local\_key'); |
|  | Định nghĩa các nghịch đảo của quan hệ One - One |  |
|  | Cách triển khai One To Many | Hãy nhớ rằng, Eloquent sẽ tự động xác định cột khóa ngoại thích hợp cho Commentmô hình. Theo quy ước, Eloquent sẽ lấy tên "trường hợp con rắn" của mô hình mẹ và tiếp nối nó với \_id. Vì vậy, trong ví dụ này, Eloquent sẽ giả sử cột khóa ngoại trên Commentmô hình là post\_id  <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  class Post extends Model  {  /\*\*  \* Get the comments for the blog post.  \*/  public function comments()  {  return $this->hasMany(Comment::class);  }  } |
|  | Định nghĩa các nghịch đảo của quan hệ One To Many |  |
|  | Cách triển khai nhiều - nhiều | posts  id - integer  name - string  videos  id - integer  name - string  tags  id - integer  name - string  taggables  tag\_id - integer  taggable\_id - integer  taggable\_type - string  <?php  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  class Mechanic extends Model  {  /\*\*  \* Get the car's owner.  \*/  public function carOwner()  {  return $this->hasOneThrough(Owner::class, Car::class);  }  } |
|  | Định nghĩa các nghịch đảo của quan hệ Many To Many |  |
|  | Cách triển khai Has Many Through | Đối số đầu tiên được truyền cho hasManyThroughphương thức là tên của mô hình cuối cùng mà chúng ta muốn truy cập, trong khi đối số thứ hai là tên của mô hình trung gian.  Mặc dù Deploymentbảng của mô hình không chứa project\_idcột, nhưng hasManyThroughquan hệ cung cấp quyền truy cập vào các triển khai của dự án thông qua $project->deployments. Để truy xuất các mô hình này, Eloquent kiểm tra project\_idcột trên Environmentbảng của mô hình trung gian . Sau khi tìm thấy các ID môi trường có liên quan, chúng được sử dụng để truy vấn Deploymentbảng của mô hình  **<?php**  namespace App\Models;  use Illuminate\Database\Eloquent\Model;  class Project extends Model  {  /\*\*  \* Get all of the deployments for the project.  \*/  public function deployments()  {  return $this->hasManyThrough(Deployment::class, Environment::class);  }  } |
|  | Quan hệ đa hình là gì | [Các mối quan hệ đa hình](https://laravel.com/docs/8.x/eloquent-relationships#polymorphic-relationships) cũng có thể được tạo ra bằng cách sử dụng các nhà máy. Các mối quan hệ đa hình "có nhiều" được tạo ra theo cùng một cách với quan hệ "có nhiều" điển hình. |
|  | Các hàm thông dụng của Eloquent | [contains](https://laravel.com/docs/8.x/eloquent-collections#method-contains)():$users->contains(1);  $users->contains(User::find(1));  Diff():  $users = $users->diff(User::whereIn('id', [1, 2, 3])->get());  Except():  $users = $users->except([1, 2, 3]);  $users = User::all();  $user = $users->find(1); [**fresh($with = [])**](https://laravel.com/docs/8.x/eloquent-collections#method-fresh) $users = $users->fresh();  $users = $users->fresh('comments'); [**intersect($items)**](https://laravel.com/docs/8.x/eloquent-collections#method-intersect) $users = $users->intersect(User::whereIn('id', [1, 2, 3])->get());  Load();  $users->load(['comments', 'posts']);  $users->load('comments.author'); [**loadMissing($relations)**](https://laravel.com/docs/8.x/eloquent-collections#method-loadMissing) $users->loadMissing(['comments', 'posts']);  $users->loadMissing('comments.author'); [**modelKeys()**](https://laravel.com/docs/8.x/eloquent-collections#method-modelKeys)**:** $users->modelKeys();  // [1, 2, 3, 4, 5] [**makeVisible($attributes)**](https://laravel.com/docs/8.x/eloquent-collections#method-makeVisible)**;** $users = $users->makeVisible(['address', 'phone\_number']); [**makeHidden($attributes)**](https://laravel.com/docs/8.x/eloquent-collections#method-makeHidden)**:** $users = $users->makeHidden(['address', 'phone\_number']); [**only($keys)**](https://laravel.com/docs/8.x/eloquent-collections#method-only)**:** $users = $users->only([1, 2, 3]); [**toQuery()**](https://laravel.com/docs/8.x/eloquent-collections#method-toquery)**:** $users = User::where('status', 'VIP')->get();  $users->toQuery()->update([  'status' => 'Administrator',  ]); [**unique($key = null, $strict = false)**](https://laravel.com/docs/8.x/eloquent-collections#method-unique) $users = $users->unique(); |
|  | hasOne dùng để làm gì ? |  |
|  | hasMany dùng để làm gì ? |  |
|  | belongsTo dùng để làm gì ? |  |
|  | belongsToMany dùng để làm gì ? |  |
|  | hasManyThrough dùng để làm gì ? |  |
|  | Cách dùng query builder để join 2 bảng |  |
|  | Cách dùng query builder để join 3 bảng |  |