LARAVEL 5.8

relaciones

27/08/2019

INTRODUCCIÓN

Las tablas de la base de datos a menudo están relacionadas entre sí. Por ejemplo, un **usuario** puede tener muchos **artículos**, o un pedido podría estar relacionado con el usuario que lo realizó. Eloquent facilita la administración y el trabajo con estas relaciones, y admite varios tipos diferentes de relaciones:

- One To One (uno a uno)
- One To Many (uno a muchos)
- Many To Many (muchos a muchos)

DEFINIENDO RELACIONES:

Las relaciones se definen como métodos en sus clases de modelo. Dado que, al igual que los modelos Eloquent, las relaciones también sirven como potentes constructores de consultas, la definición de relaciones como métodos proporciona potentes capacidades de consulta y encadenamiento de métodos.

UNO A UNO:

Una relación uno a uno es una relación muy básica. Por ejemplo, un modelo de **usuario** podría estar asociado con un **teléfono**. Para definir esta relación, colocamos un método de teléfono en el modelo de Usuario. El método del teléfono debe llamar al método **hasOne** y devolver su resultado:

```
<?php
```

```
namespace App;

use Illuminate\Database\Eloquent\Model;


class User extends Model
{
    public function phone()
    {
        return $this->hasOne('App\Phone);
    }
}
```

El primer argumento pasado al método hasOne es el nombre del modelo relacionado. Una vez que se define la relación, podemos recuperar el registro relacionado utilizando las propiedades dinámicas de Eloquent. Las propiedades dinámicas le permiten acceder a los métodos de relación como si fueran propiedades definidas en el modelo:

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

Eloquent determina la llave foránea de la relación en función del nombre del modelo. En este caso, se supone automáticamente que el modelo de teléfono tiene una llave foránea **id_user**. Si desea anular esta convención, puede pasar un segundo argumento al método **hasOne**:

```
return $this->hasOne('App\Phone', 'foreign_key');
```

DEFINIENDO RELACIÓN INVERSA:

Podemos acceder al modelo de teléfono desde nuestro usuario. Ahora, definamos una relación en el modelo de teléfono que nos permitirá acceder al usuario que posee el teléfono. Podemos definir el inverso de una relación **hasOne** usando el método **belongTo**:

```
<?php

namespace App;
use Illuminate\Database\Eloquent\Model;

class Phone extends Model
{
    public function user()
    {
        return $this->belongsTo('App\User');
    }
}
```

UNO A MUCHOS:

Una relación de uno a muchos se utiliza para definir relaciones donde un solo modelo posee cualquier cantidad de otros modelos. Por ejemplo, una publicación de blog puede tener una cantidad infinita de comentarios. Las relaciones uno a muchos se definen colocando una función en su modelo:

```
<?php

namespace App;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    public function comments()
```

```
    {
        return $this->hasMany('App\Comment');
    }
}
```

Una vez que se ha definido la relación, podemos acceder a la colección de comentarios accediendo a la propiedad **comments**. Recuerde, dado que Eloquent proporciona "propiedades dinámicas", podemos acceder a los métodos de relación como si estuvieran definidos como propiedades en el modelo:

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

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