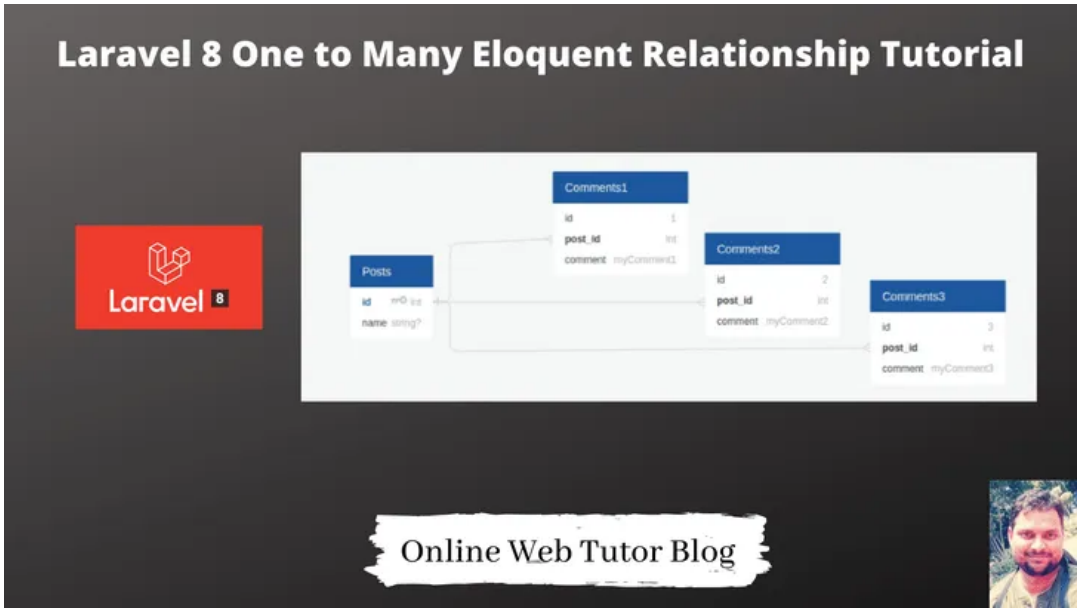

Tutoriel sur les relations éloquentes Laravel 8 One to Many

1 avril 2021 par Sanjay Kumar

Table des matières

Partagez cet article

La relation éloquente de Laravel est une fonctionnalité très importante qui relie une ou plusieurs tables dans une chaîne. C'est le substitut des jointures dans laravel.

Laravel fournit ces relations suivantes -

- **Un par un**
- **Un à plusieurs**
- **Plusieurs à plusieurs**
- Un à plusieurs (inverse) / Appartient à
- **A un travers**
- **A beaucoup à travers**

Les relations Eloquent sont définies comme des méthodes sur vos classes de modèle Eloquent. Dans cet article, nous verrons le concept de relation laravel 8 One to Many Eloquent ainsi que nous mettrons en œuvre l'inverse de la relation un à plusieurs, c'est-à-dire appartient à.

Cet article vous donnera le concept détaillé de la mise en œuvre d'une relation un à plusieurs dans laravel.

Commençons.

Installation de l'application Laravel

L'installation de Laravel peut se faire de deux manières.

- Installateur Laravel
- En utilisant le compositeur

Installateur Laravel

Pour installer Laravel via le programme d'installation de Laravel, nous devons d'abord installer son programme d'installation. Nous devons utiliser le compositeur pour cela.

```
$ composer global nécessite laravel/installer
```

Cette commande installera le programme d'installation de laravel sur le système. Cette installation est à portée globale, vous tapez donc la commande à partir de n'importe quel répertoire du terminal. Pour vérifier, tapez la commande donnée -

```
$ laravel
```

Cette commande ouvrira une palette de commandes de Laravel Installer.

Pour créer un projet ad install laravel dans le système,

```
$ laravel nouveau blog
```

Avec le nom de **blog,** un projet laravel sera créé à votre chemin spécifié.

En utilisant le compositeur

Si votre système n'a pas installé composer, **Apprenez les étapes d'installation de composer** .

Voici la commande complète pour créer un projet laravel-

```
$ composer create-project --prefer-dist blog laravel/laravel
```

Après avoir suivi ces étapes, nous pouvons installer une application Laravel dans le système.

To start the development server of Laravel –

```
$ php artisan serve
```

This command outputs –

Starting Laravel development server: http://127.0.0.1:8000

Assuming laravel already installed at system.

Create Database & Connect

To create a database, either we can create via Manual tool of PhpMyadmin or by means of a mysql command.

```
CREATE DATABASE laravel_app;
```

To connect database with application, Open **.env file** from application root. Search for **DB_** and update your details.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_app
DB_USERNAME=root
DB_PASSWORD=root
```

Create Migrations

- Migration file for Posts table
- Migration file for Comments table

Open project into terminal and run these migration command.

$ php artisan make:migration CreatePostsTable

$ php artisan make:migration CreateCommentsTable

It will create two files 2021_04_01_162614_**create_posts_table.php** & 2021_04_01_162630_**create_comments_table.php** at /database/migrations folder.

Open 2021_04_01_162614_**create_posts_table.php** and write this complete code into it.

```
#PostsMigration
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreatePostsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('posts', function (Blueprint $table) {
            $table->id();
            $table->string("title");
            $table->text("description")->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('posts');
    }
}
```

Open 2021_04_01_162630_**create_comments_table.php** and write this code into it.

```
#CommentsMigration
<?php
```

```
use Illuminate\Support\Facades\Schema;

class CreateCommentsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('comments', function (Blueprint $table) {
            $table->id();
            $table->unsignedInteger('post_id');
            $table->string("comment");
            $table->timestamps();
            $table->foreign('post_id')->references('id')->on('posts')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('comments');
    }
}
```

Run Migrations

Next, we need to create tables inside database.

```
$ php artisan migrate
```

This command will create tables inside database.

Create Model

Next, we need to create two models. Back to terminal and run these artisan commands.

```
$ php artisan make:model Post
```

```
$ php artisan make:model Comment
```

Open **Post.php** and write this complete code into it.

Post.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\Comment;

class Post extends Model
{
    use HasFactory;

    protected $fillable = ["title", "description"];

    /**
     * Get the comments for the blog post.
     */
    public function comments()
    {
        return $this->hasMany(Comment::class);
    }
}
```

$this->hasMany(Comment::class); It is creating one to many relationship.

Open **Comment.php** and write this complete code into it.

Comment.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use App\Models\Post;

class Comment extends Model
{
    use HasFactory;

    protected $fillable = ["post_id", "comment"];

    /**
     * Get the post that owns the comment.
     */
    public function post()
    {
        return $this->belongsTo(Post::class);
    }
}
```

$this->belongsTo(Post::class); It is creating inverse of one to many relationship.

Open mysql and run these queries to insert dummy data into posts and comments table.

Data for Posts Table

```
#Query
--
-- Dumping data for table `posts`
--

INSERT INTO `posts` (`id`, `title`, `description`, `created_at`, `updated_at`) VALUES
(1, 'Odie Robel', 'Quae eveniet hic qui ut molestias dignissimos. Corporis dolore dolor eius illum quia
(2, 'Earl Weissnat', 'Quaerat quasi libero et error commodi. Nihil consectetur suscipit deleniti corpor
(3, 'Ms. Samanta Haley MD', 'Sit ut architecto sit quis. Non ipsa sed distinctio ullam vel. Minus aut r
(4, 'Mr. Bertha Jakubowski III', 'Nemo recusandae voluptas quae quisquam dolores quia. Quo totam nostru
(5, 'Dr. Casimir D\'Amore DVM', 'Sed natus voluptatibus non blanditiis porro. Totam veniam officia dolo
(6, 'Ms. Letitia Koch Sr.', 'Aut ut quidem nemo qui pariatur amet itaque. Minima dicta enim quia volupt
(7, 'Sophie Rempel', 'Consequatur sed harum ut similique. Voluptate nisi nihil illum eum dolore. Et qui
(8, 'Dr. Virginia Gislason', 'Et placeat aut consequatur. Est dolorum et ut non minus. Ut ab et alias e
(9, 'Miss Caitlyn Heller', 'Est debitis vitae qui a a. Commodi in ut corrupti nobis repellendus libero
(10, 'Mozell Moore', 'Eos enim molestias voluptate quo ipsa. Autem et voluptas quaerat est maxime. Magn
```

Data for Comments Table

```
#Query
--
-- Dumping data for table `comments`
--

INSERT INTO `comments` (`id`, `post_id`, `comment`, `created_at`, `updated_at`) VALUES
(1, 1, 'Exercitationem ea possimus saepe. Odit quas ut ea vitae officiis. Consequatur vel beatae aut qu
(2, 2, 'Voluptatem repellendus alias hic tempora corrupti dolor sit expedita. Qui quasi pariatur sint c
(3, 1, 'Aut consequatur ipsum nisi quo et ea deserunt. Rem qui ipsa sed nisi. Similique vel voluptatem
(4, 4, 'Odit hic aut ut molestiae quod. Tempora harum est ut sit ratione et. Quam vitae explicabo fugit
(5, 3, 'Cum corporis assumenda adipisci rem deleniti. Ut sint voluptate et qui saepe fugiat in. Laborum
(6, 3, 'Nihil explicabo nisi eveniet aut provident. Odio molestias dolores sint quia omnis dolore harum
(7, 7, 'Autem id omnis quos. Et eos et eos expedita ad libero. Quos quia labore quibusdam maxime cupidi
(8, 2, 'Alias placeat expedita in laudantium. Quos ea qui voluptatibus rem. Temporibus dignissimos
(9, 8, 'Ut dolorem quasi hic cumque. Dolorum eum porro reiciendis et neque. Quis praesentium quis iu
(10, 8, 'Incidunt ab explicabo veritatis et. Perspiciatis eum incidunt officia aut autem impedit vi
```

Ouvrez n'importe quel contrôleur, par exemple le fichier **SiteController.php** , nous avons créé deux méthodes dans lesquelles nous avons utilisé des méthodes de modèle en tant que propriété.

```
#Manette
< ? php

espace de noms  App\Http\Controllers ;

utilisez  Illuminate\Http\Request ;
utilisez  App\Models\Post ;
utilisez  App\Models\Comment ;

la classe  SiteController  étend le  contrôleur
{
     fonction  publique getComments ( $post_id )
    {
        // Passage de l'identifiant du message dans find()
        return  Post :: trouver ( $post_id ) -> commentaires ;
    }

     fonction  publique getPost ( $comment_id )
    {
        // Passage de l'identifiant du commentaire dans find()
        return  Commentaire :: trouver ( $comment_id ) -> poster ;
    }
}
```

- **Post ::find($post_id)->commentaires ;** Il trouvera les valeurs détaillées des commentaires par identifiant de publication. **Un à plusieurs**
- **Commentaire ::find($comment_id)->poster ;** Il trouvera les détails de la publication par identifiant de commentaire. **Inverse de Un à Plusieurs / Appartient à**

Créer des itinéraires

Ouvrez **web.php** à partir du dossier /routes et ajoutez-y ces routes.

```
web.php
# Ajouter ceci à l'en-tête
utilisez  App\Http\Controllers\SiteController ;

Route :: get ( 'get-comments/{id}' , [ SiteController :: class , 'getComments' ]);
Route :: get ( 'get-post/{id}' , [ SiteController :: class , 'getPost' ]);
```

Test d'applications

Ouvrez le projet sur le terminal et tapez la commande pour démarrer le serveur de développement

Obtenir des commentaires - http://127.0.0.1:8000/get-comments/1

Obtenir les détails de la publication - http://127.0.0.1:8000/get-post/1

Nous espérons que cet article vous a aidé à en savoir plus sur Laravel 8 One to Many Eloquent Relationship Tutorial de manière très détaillée.

If you liked this article, then please subscribe to our **YouTube Channel** for PHP & it's framework, WordPress, Node Js video tutorials. You can also find us on **Twitter** and **Facebook**.

Find More on Laravel 8 Articles here

- How to Create Multi Language Website in Laravel 8
- How To Read XML File in Laravel 8 – Example
- How To Upload And Save XML Data in Laravel 8
- Laravel 8 Ajax Post Request Tutorial
- Laravel 8 Authentication using Jetstream with Inertia Js
- Laravel 8 Authentication using Jetstream with Livewire
- Laravel 8 Authentication with Breeze Tutorial
- Laravel 8 Clear Cache of Route, View & Config
- Laravel 8 Cron Job Task Scheduling Tutorial
- Laravel 8 DataTable Ajax Pagination with Search And Sort
- Laravel 8 Firebase Push Notification Tutorial
- Laravel 8 Form Validation Methods
- Guide d'installation de Laravel 8 – Framework PHP
- Guide complet des mises en page et des vues de Laravel 8
- Tutoriel de routage Laravel 8 Guide étape par étape
- Laravel 8 Envoyer un courrier à l'aide du serveur SMTP Gmail
- Laravel 8 Envoyer une notification push à Android à l'aide de Firebase
- Laravel 8 Envoyer une notification push à IOS à l'aide de Firebase
- Personnalisation du Stub Laravel 8

Articles Similaires:

- Laravel 8 Tutoriel sur les relations éloquentes de plusieurs à plusieurs
- Laravel 8 Tutoriel sur les relations éloquentes de Laravel 8
- Laravel 8 en a beaucoup grâce au didacticiel sur les relations éloquentes
- Laravel 8 a un tutoriel sur les relations éloquentes
- Tutoriel sur les mutateurs et accesseurs éloquents dans Laravel 8
- Journal de magasin Laravel 8 des requêtes SQL éloquentes

Laravel 8
‹ Laravel 8 Tutoriel sur les relations éloquentes de Laravel 8
› Laravel 8 Tutoriel sur les relations éloquentes de plusieurs à plusieurs

Retrouvez-nous sur Youtube

Tuteur Web en ligne Youtube

Publicité

Articles populaires

Pardon. Pas de données jusqu'à présent.

Catégories

- Bloguer (1)
- GâteauPHP 4 (11)
- CodeIgniter 4 (156)
- Éducation (1)
- jQuery et Javascript (39)
- Laravel 8 (152)
- MySQL (dix)
- Noeud J (15)
- Divers PHP (33)
- WordPress (22)

Messages récents

Menu

Menu