



## Laravel 8 a un tutoriel sur les relations éloquentes

3 mai 2021 par Sanjay Kumar

### Table des matières

1. Installation de l'application Laravel
2. Créer une base de données et se connecter
3. Créer des migrations
4. Créer un modèle
5. Exemple d'insertion de données
6. Calling Method at Controller
7. Créer des itinéraires
8. Test d'applications

### Partagez cet article

Temps de lecture : 7 minutes 4 887 Vues

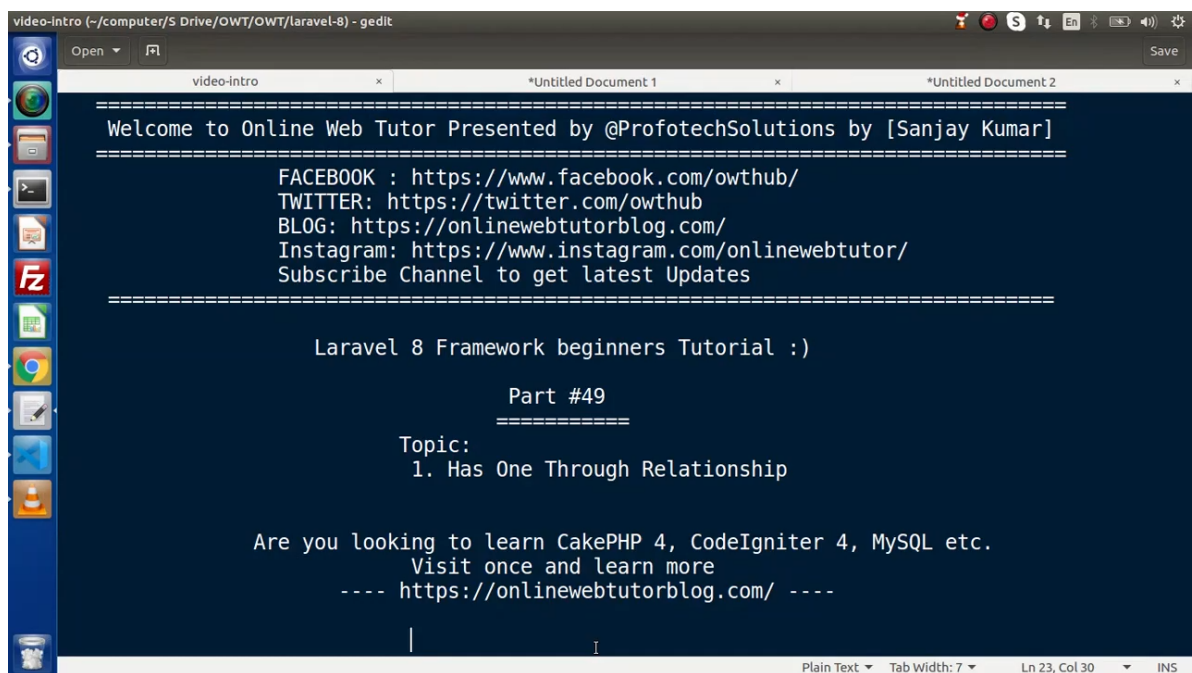
La relation éloquente de Laravel est une fonctionnalité très importante qui relie une ou plusieurs tables dans une chaîne. C'est le substitut des jointures dans laravel.

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 laravel 8 Has One à travers une relation éloquente.

Cet article vous donnera le concept détaillé de la mise en œuvre de Has One via une relation dans laravel.



```
video-intro (~/.computer/S Drive/OWT/OWT/laravel-8) - gedit
=====
Welcome to Online Web Tutor Presented by @ProfotechSolutions by [Sanjay Kumar]
=====
FACEBOOK : https://www.facebook.com/owthub/
TWITTER: https://twitter.com/owthub
BLOG: https://onlinewebtutorblog.com/
Instagram: https://www.instagram.com/onlinewebtutor/
Subscribe Channel to get latest Updates
=====

Laravel 8 Framework beginners Tutorial :)

Part #49
=====
Topic:
1. Has One Through Relationship

Are you looking to learn CakePHP 4, CodeIgniter 4, MySQL etc.
Visit once and learn more
---- https://onlinewebtutorblog.com/ ----
```

For this tutorial we will consider a **people table**, **brokers table** and **homes table**. This means a person will contact a broker to take a home

## Personne -> Courtier -> Domicile

Commençons.

## Installation de l'application Laravel

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

- Installeur Laravel
- En utilisant le compositeur

### Installeur 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

Alternativement, nous pouvons également installer Laravel par la commande Composer **create-project** .

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.

Pour démarrer le serveur de développement de Laravel –

```
$ php service artisanal
```

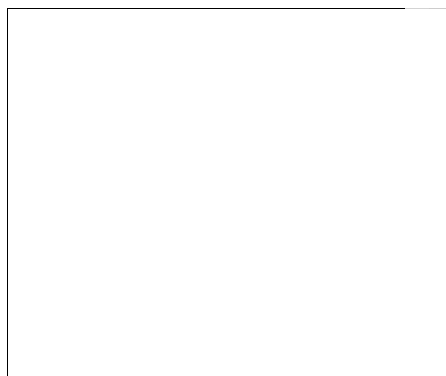
Cette commande affiche –

Démarrage du serveur de développement Laravel : http://127.0.0.1:8000

**En supposant que laravel est déjà installé sur le système.**

## Créer une base de données et se connecter

Pour créer une base de données, soit nous pouvons créer via l'outil manuel de PhpMyadmin soit au moyen d'une commande mysql.



CRÉER UNE BASE DE DONNÉES laravel\_app ;

Pour connecter la base de données à l'application, **ouvrez le fichier .env** à partir de la racine de l'application. Recherchez **DB\_** et mettez à jour vos informations.

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

## Créer des migrations

Nous avons besoin de quelques fichiers de migration -

- Migration des personnes pour stocker les données des personnes
- Migration des courtiers vers les courtiers en magasin
- Home Migration pour stocker les données des maisons

Ouvrez le projet dans le terminal et exécutez ces commandes artisanales.

```
$ php artisan make:migration create_people_table
```

```
$ php artisan make:migration create_brokers_table
```

```
$ php artisan make:migration create_homes_table
```

Il créera deux fichiers de migration 2021\_05\_03\_162204\_create\_people\_table.php , 2021\_05\_03\_162246\_create\_brokers\_table.php et 2021\_05\_03\_162253\_create\_homes\_table.php à l'emplacement /database/migrations .

Ouvrez 2021\_05\_03\_162204\_create\_people\_table.php et écrivez-y ce code complet.

Migrer #1

< ? php

```
utilisez Illuminate\Database\Migrations\Migration ;
utilisez Illuminate\Database\Schema\Blueprint ;
utilisez Illuminate\Support\Facades\Schema ;

la classe CreatePeopleTable étend la migration
{
    /**
     * Exécutez les migrations.
     *
     * @return void
     */
    function publique ( )
    {
        Schéma :: créer ( 'personnes' , fonction ( Blueprint $table ) {
            $table -> identifiant ();
            $table -> chaîne ( "nom" , 120 );
            $table -> chaîne("email", 120)->unique();
            $table -> horodatages ();
        });
    }

    /**
     * Inverser les migrations.
     *
     * @return void
     */
    function publique vers le bas ( )
    {
        Schéma :: dropIfExists ( 'personnes' );
    }
}
```

```
}
}
```

Ouvrez 2021\_05\_03\_162246\_**create\_brokers\_table.php** et écrivez-y ce code.

Migrer #2

```
< ? php
```

```
utilisez Illuminate\Database\Migrations\Migration ;
utilisez Illuminate\Database\Schema\Blueprint ;
utilisez Illuminate\Support\Facades\Schema ;

la classe CreateBrokersTable étend la migration
{
    /**
     * Exécutez les migrations.
     *
     * @return void
     */
    fonction publique ( )
    {
        Schéma ::create('brokers', function (Blueprint $table) {
            $table -> identifiant ();
            $table -> chaîne ( "nom" , 120 );
            $table -> chaîne ( "email" , 120 ) -> unique ();
            $table -> unsignedBigInteger ( 'person_id' );
            $table -> horodatages ();
            $table -> étranger ( 'person_id' ) -> références ( 'id' ) -> on ( 'people' );
        });
    }

    /**
     * Inverser les migrations.
     *
     * @return void
     */
    fonction publique vers le bas ( )
    {
        Schéma :: dropIfExists ( 'brokers' );
    }
}
```

Ouvrez 2021\_05\_03\_162253\_**create\_homes\_table.php** et écrivez-y ce code complet.

Migrer #3

```
< ? php
```

```
utilisez Illuminate\Database\Migrations\Migration ;
utilisez Illuminate\Database\Schema\Blueprint ;
utilisez Illuminate\Support\Facades\Schema ;

la classe CreateHomesTable étend la migration
{
    /**
     * Exécutez les migrations.
     *
     * @return void
```

```

*/
function publique ( )
{
    Schema :: create ( 'homes' , function ( Blueprint $table ) {
        $table -> id();
        $table -> chaîne ( "titre" , 155 );
        $table -> texte ( "emplacement" );
        $table -> unsignedBigInteger ( 'broker_id' );
        $table -> horodatages ();
        $table -> étranger ( 'broker_id' ) -> références ( 'id' ) -> on ( 'brokers' );
    });
}

/**
 * Inverser les migrations.
 *
 * @retour vide
 */
function publique vers le bas ()
{
    Schéma :: dropIfExists ( 'homes' );
}
}

```

## Exécuter les migrations

Ensuite, nous devons créer des tables dans la base de données.

```
$ php artisan migrer
```

Cette commande créera des tables dans la base de données.

## Créer un modèle

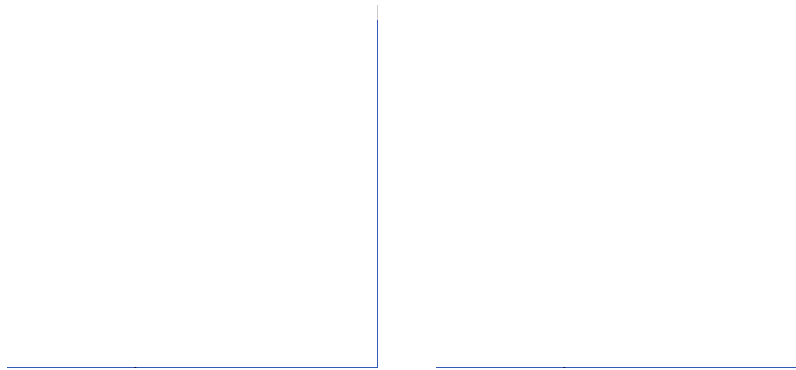
Ensuite, nous devons créer trois modèles. Retournez au terminal et exécutez ces commandes artisanales.

```
$ php artisan make:model Person
```

```
$ php artisan make:model Broker
```

```
$ php artisan make:model Accueil
```

Ces commandes créeront trois fichiers **Person.php**, **Broker.php** et **Home.php** dans le dossier /app/Models.



Ouvrez **Person .php** et écrivez-y ce code complet.

Personne.php

```
< ? php
```

```
namespace App\Models;
```

```
utilisez Illuminate\Database\Eloquent\Factories\HasFactory ;
```

```
utilisez Illuminate\Database\Eloquent\Model ;
```

```
utilisez App\Models\Home ;
```

```
utilisez App\Models\Broker ;
```

```
la classe Personne étend le modèle
```

```
{
```

```
    utilisez HasFactory ;
```

```
    fonction publique homeInformation ()
```

```
    {
```

```
        return $this -> hasOneThrough ( Home :: class , Broker :: class );
```

```
    }
```

```
}
```

**\$this->hasOneThrough(Home::class, Broker::class);** This line is implementing the relationship i.e Has One Through

Reste pour tous les modèles comme pour **Broker.php** et le code **home.php** sera le même que le code par défaut.

## Exemple d'insertion de données



Ouvrez mysql et exécutez ces requêtes pour insérer des données factices dans les tables people, brokers et homes.

### Tableau des données pour les personnes

```
#PeopleTable

--
-- Vidage des données pour la table `people`
--

INSERT INTO `personnes` ( `id` , ` name` , `email` , `created_at` , `updated_at` ) VALEURS
( 1 , 'Sanjay Kumar' , 'sanjay@gmail.com' , NULL , NULL ),
(2, 'Ashish Kumar', 'ashish@gmail.com', NULL, NULL);
```

### Tableau des données pour les courtiers

```
#BrokersTable

--
-- Dumping des données pour la table `brokers`
--

INSERT INTO `brokers` ( `id` , ` name` , `email` , `person_id` , `created_at` , `updated_at` ) VALEUR
( 1 , 'Courtier 1' , 'courtier1@gmail.com' , 1 , NULL , NULL ),
( 2 , 'Courtier 2' , 'courtier2@gmail.com' , 2 , NULL , NULL );
```

### Tableau des données pour les foyers

```
#HomesTable

--
-- Dumping des données pour la table `homes`
--

INSERT INTO `homes` ( `id` , `title` , `location` , `broker_id` , `created_at` , `updated_at` ) VALEU
( 1 , 'Maison 1' , 'Emplacement 1, Rue 1, Exemple' , 2 , NULL , NULL ),
( 2 , 'Maison 2' , 'Emplacement 2, Rue 2, Exemple ' , 1 , NULL , NULL );
```

## Méthode d'appel au contrôleur

Ouvrez n'importe quel contrôleur, par exemple le fichier **SiteController.php**, nous avons créé une méthode dans laquelle nous avons utilisé la méthode du modèle en tant que propriété.

```
#Manette

< ? php

espace de noms App\Http\Controllers ;

utilisez Illuminate\Http\Request ;
utilisez App\Models\Person ;

la classe SiteController étend le contrôleur
{
    // Pour obtenir les détails de la maison d'une personne
    fonction publique getHome ()
    {
        return Person :: find ( 1 ) -> homeInformation;
    }
}
```

**Personne ::find(1)->homeInformation** ; Il renverra les informations d'accueil de la personne par son ID.

## Créer des itinéraires

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

```
web.php

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

Route :: get ( 'home-information' , [ SiteController :: class , 'getHome' ] );
```

## Test d'applications

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

```
$ php service artisanal
```

**Obtenir des informations sur la maison par ID de personne** - <http://127.0.0.1:8000/home-information>

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

Si vous avez aimé cet article, veuillez vous abonner à notre [chaîne YouTube](#) pour PHP et ses tutoriels vidéo sur le Framework, WordPress, Node Js. Vous pouvez également nous retrouver sur [Twitter](#) et [Facebook](#).

### En savoir plus sur les articles de Laravel 8 ici

- [Comment créer un site Web multilingue dans Laravel 8](#)
- [Comment lire un fichier XML dans Laravel 8 - Exemple](#)
- [Comment télécharger et enregistrer des données XML dans Laravel 8](#)
- [Tutoriel de demande de publication Laravel 8 Ajax](#)
- [Authentification Laravel 8 à l'aide de Jetstream avec Inertia Js](#)
- [Authentification Laravel 8 à l'aide de Jetstream avec Livewire](#)
- [Tutoriel d'authentification Laravel 8 avec Breeze](#)
- [Laravel 8 Effacer le cache de la route, de la vue et de la configuration](#)
- [Tutoriel de planification des tâches de la tâche Laravel 8 Cron](#)
- [Laravel 8 DataTable Ajax Pagination avec recherche et tri](#)
- [Tutoriel de notification push de Laravel 8 Firebase](#)
- [Méthodes de validation des formulaires Laravel 8](#)
- [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 en a beaucoup grâce au didacticiel sur les relations éloquentes](#)
- [Laravel 8 Tutoriel sur les relations éloquentes de Laravel 8](#)
- [Tutoriel sur les relations éloquentes Laravel 8 One to Many](#)
- [Laravel 8 Tutoriel sur les relations éloquentes de plusieurs à plusieurs](#)
- [Tutoriel sur les mutateurs et accesseurs éloquents dans Laravel 8](#)
- [Journal de magasin Laravel 8 des requêtes SQL éloquentes](#)

#### 📁 Laravel 8

- ◀ [Comment se connecter aux données utilisateur dans le didacticiel Laravel 8](#)
- ▶ [Laravel 8 en a beaucoup grâce au didacticiel sur les relations éloquentes](#)

### Advertisement



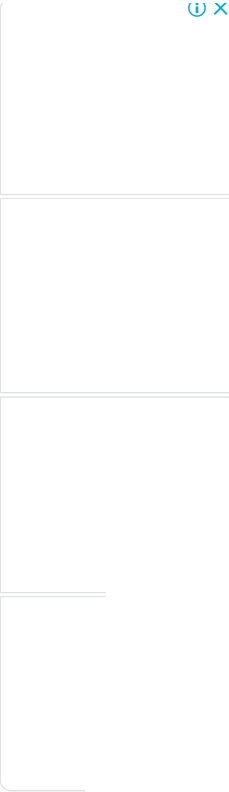
[signaler cette annonce](#)

## Retrouvez-nous sur Youtube



Tuteur Web en ligne Youtube

## Publicité



 **ezoic**  
signaler cette annonce

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 \(22\)](#)

## Publicité



signaler cette annonce

## Messages récents

[Comment lire des données XML en JSON dans CodeIgniter 4](#)

[Comment lire des données XML en JSON dans le didacticiel PHP](#)

[Somme MySQL avec le tutoriel de condition If](#)

[Quels sont les composants fonctionnels de CodeIgniter 4 ?](#)

[Concept d'image de chargement paresseux à l'aide de jQuery](#)



signaler cette annonce















