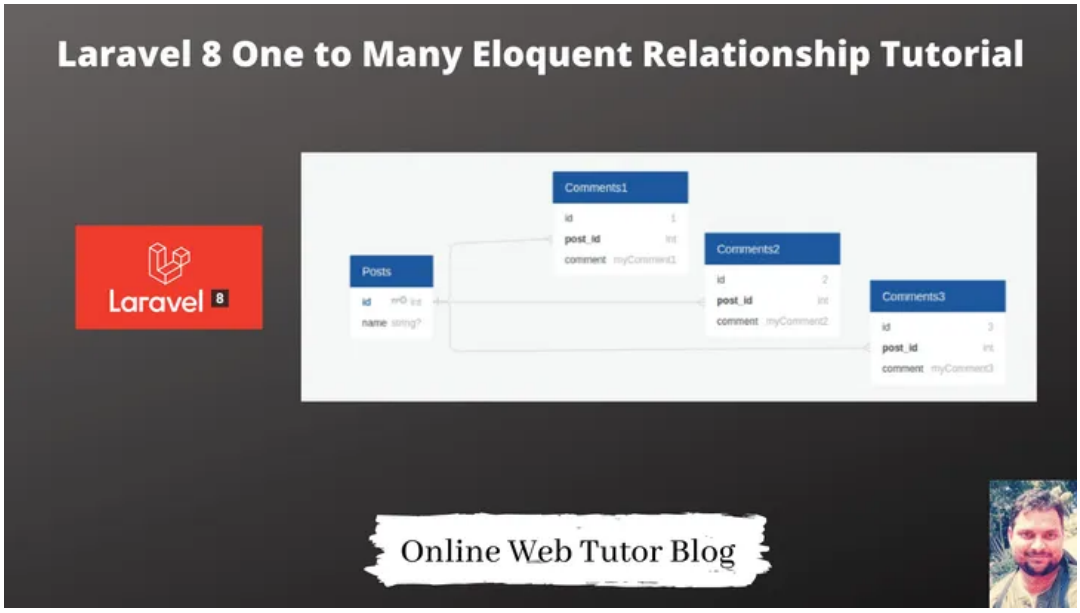

Laravel 8 One to Many Eloquent Relationship Tutorial

April 1, 2021 by Sanjay Kumar

Table of Contents

Share this Article

Laravel eloquent relationship is a very important feature which connects one or more tables in a chain. This is the substitute of joins in laravel.

Laravel provides these following relationships –

- One To One
- **One To Many**
- Many To Many
- One To Many (Inverse) / Belongs To
- Has One Through
- Has Many Through

Eloquent relationships are defined as methods on your Eloquent model classes. Inside this article we will see the concept of laravel 8 One to Many Eloquent relationship as well as we will implement inverse of one to many relationship i.e belongs to.

This article will give you the detailed concept of about implementation of one to many relationship in laravel.

Let's get started.

Installation of Laravel Application

Laravel Installation can be done in two ways.

- Laravel Installer
- By using composer

Laravel Installer

To install Laravel via Laravel installer, we need to install it's installer first. We need to make use of composer for that.

```
$ composer global require laravel/installer
```

This command will install laravel installer at system. This installation is at global scope, so you type command from any directory at terminal. To verify type the given command –

```
$ laravel
```

This command will open a command palette of Laravel Installer.

To create ad install laravel project in system,

```
$ laravel new blog
```

With the name of **blog** a laravel project will be created at your specified path.

By using composer

Alternatively, we can also install Laravel by Composer command **create-project**.

Here is the complete command to create a laravel project-

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

After following these steps we can install a Laravel application into system.

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

We need to create two migration files.

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\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
```

```
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
```

This command will create two files **Post.php** & **Comment.php** at /app/Models folder.

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 in i
(9, 8, 'Ut dolorem quasi hic cumque. Dolorum eum porro reiciendis et neque. Quis praesentium quis i
(10, 8, 'Incidunt ab explicabo veritatis et. Perspiciatis eum incidunt officia aut autem impedit vi
```

Open any controller say **SiteController.php** file, we have created two methods in which we used model methods as a property.

```
#Controller
<?php

namespace App\Http\Controllers;

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

class SiteController extends Controller
{
    public function getComments($post_id)
    {
        // Passing post id into find()
        return Post::find($post_id)->comments;
    }

    public function getPost($comment_id)
    {
        // Passing comment id into find()
        return Comment::find($comment_id)->post;
    }
}
```

- **Post::find($post_id)->comments;** It will find comments detail values by post id. **One to Many**
- **Comment::find($comment_id)->post;** It will find post detail by comment id. **Inverse of One to Many / Belongs To**

Create Routes

Open **web.php** from /routes folder and add these routes into it.

```
web.php
# Add this to header
use App\Http\Controllers\SiteController;

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

Application Testing

Open project to terminal and type the command to start development server

```
$ php artisan serve
```

Get Post detail– http://127.0.0.1:8000/get-post/1

We hope this article helped you to learn about Laravel 8 One to Many Eloquent Relationship Tutorial in a very detailed way.

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
- Laravel 8 Installation Guide – PHP Framework
- Laravel 8 Layouts And Views Complete Guide
- Laravel 8 Routing Tutorial Step by Step Guide
- Laravel 8 Send Mail using Gmail SMTP Server
- Laravel 8 Send Push Notification to Android Using Firebase
- Laravel 8 Send Push Notification to IOS Using Firebase
- Laravel 8 Stub Customization

Related Posts:

- Laravel 8 Many to Many Eloquent Relationship Tutorial
- Laravel 8 One to One Eloquent Relationship Tutorial
- Laravel 8 Has Many Through Eloquent Relationship Tutorial
- Laravel 8 Has One Through Eloquent Relationship Tutorial
- Eloquent Mutators and Accessors in Laravel 8 Tutorial
- Laravel 8 Store Log Of Eloquent SQL Queries

Laravel 8
Laravel 8 One to One Eloquent Relationship Tutorial
Laravel 8 Many to Many Eloquent Relationship Tutorial

Find Us on Youtube

Online Web Tutor Youtube

Advertisement

Popular Posts

Sorry. No data so far.

Categories

- Blogging (1)
- CakePHP 4 (11)
- CodeIgniter 4 (156)
- Education (1)
- jQuery & Javascript (39)
- Laravel 8 (152)
- MySQL (10)
- Node Js (15)
- PHP Miscellaneous (33)
- Wordpress (22)

Recent Posts