Rakamin Academy

investree

Virtual Internship Experience

Laravel: Eloquent

Laravel : Eloquent Object Relational Mapping (ORM)

Eloquent ORM merupakan fitur yang dimiliki oleh laravel yang berisi perintah *active record*
(query sql) yang digunakan untuk mengelola *database*

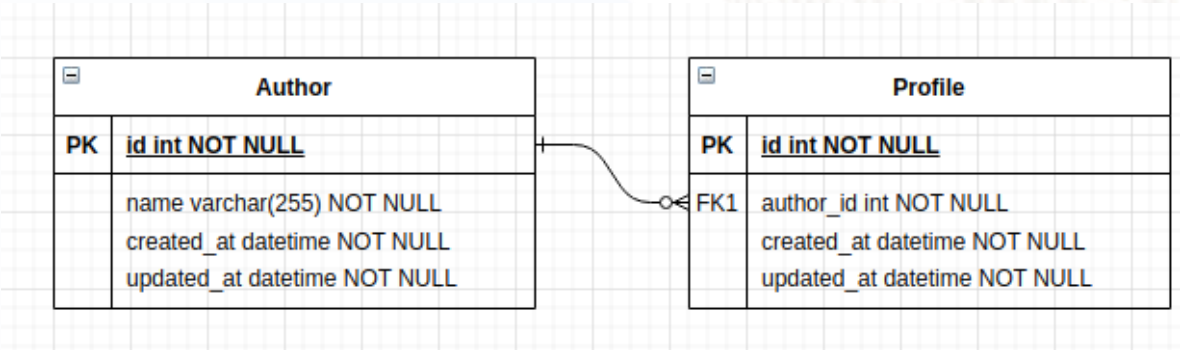
Dengan fitur Eloquent ORM ini kita dapat dengan mudah menjalankan perintah seperti *insert, update, delete*

Fitur ini juga mempermudah kita untuk membuat relasi antar *table* melalui model

Pada pembahasan kali ini kita akan membahas berbagai macam bentuk relasi yang terdapat dalam Eloquent ORM

One to One

Merupakan *table* yang memiliki 1 relasi ke *table* lainnya, dapat dilihat pada contoh diagram dibawah ini



Kita sudah memiliki 2 model yaitu Author & Profile dimana terdapat relasi *foreign key* author_id pada *table profiles* yang berelasi dengan id pada *table authors*, kemudian kita bisa membuat kedua model tersebut pada *project laravel*

Jalankan kedua perintah ini

```
php artisan make:model Author -m
```

Kemudian jalankan perintah ini

```
php artisan make:model Profile -m
```

Kemudian ubah isi file migrations table authors seperti dibawah ini

2021_12_23_175504_create_authors_table.php

database > migrations > 2021_12_23_175504_create_authors_table.php > PHP Intelephense > CreateAuthorsTable > down

```php
<?php

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

class CreateAuthorsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('authors', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

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

Dan ubah isi file *migrations table profiles* seperti dibawah ini

2021_12_23_175512_create_profiles_table.php

database > migrations > 2021_12_23_175512_create_profiles_table.php > PHP Intelephense > CreateProfilesTable > up

```php
class CreateProfilesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('profiles', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('author_id')->unique();
            $table->foreign('author_id')->references('id')->on('authors');
            $table->timestamps();
        });
    }

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

Dan jalankan perintah *migrate* untuk melakukan insert pada kedua *file migration* tersebut

Langkah selanjutnya ubah isi model app/Models/Author.php menjadi seperti ini

Author.php

app > Models > Author.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Author extends Model
{
    use HasFactory;

    protected $guarded = ['id'];

    public function profile()
    {
        return $this->hasOne(Profile::class);
    }
}
```

Pada bagian model diatas terdapat *property* `protected $guarded = ['id'];` merupakan fitur *mass assignment* yang terdapat pada laravel yang berfungsi mengirimkan data dalam bentuk array dengan *array_keys* adalah kolom yang digunakan pada *database*

Untuk dokumentasi detail mengenai *mass assignment* dapat di cek melalui *link* dibawah ini

https://laravel.com/docs/8.x/eloquent#mass-assignment

Dan terdapat *method profile()* yang merupakan penamaan yang memiliki relasi ke model Profile
& method hasOne digunakan untuk mendefinisikan relasi 1 *to* 1 dimana parameter pertama adalah model yang memiliki relasi dan parameter kedua adalah *foreign key* dari *table profiles*

Kemudian isi ubah file app/Models/Profile.php menjadi seperti dibawah ini

Profile.php

app > Models > Profile.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Profile extends Model
{
    use HasFactory;

    protected $guarded = ['id'];

    public function author()
    {
        return $this->belongsTo(Author::class);
    }
}
```

Dan terdapat method *author()* yang digunakan untuk membuat relasi ke model Author
Kemudian terdapat fungsi *belongsTo* mendefinisikan kebalikan dari hubungan *hasOne* menggunakan metode *belongsTo*

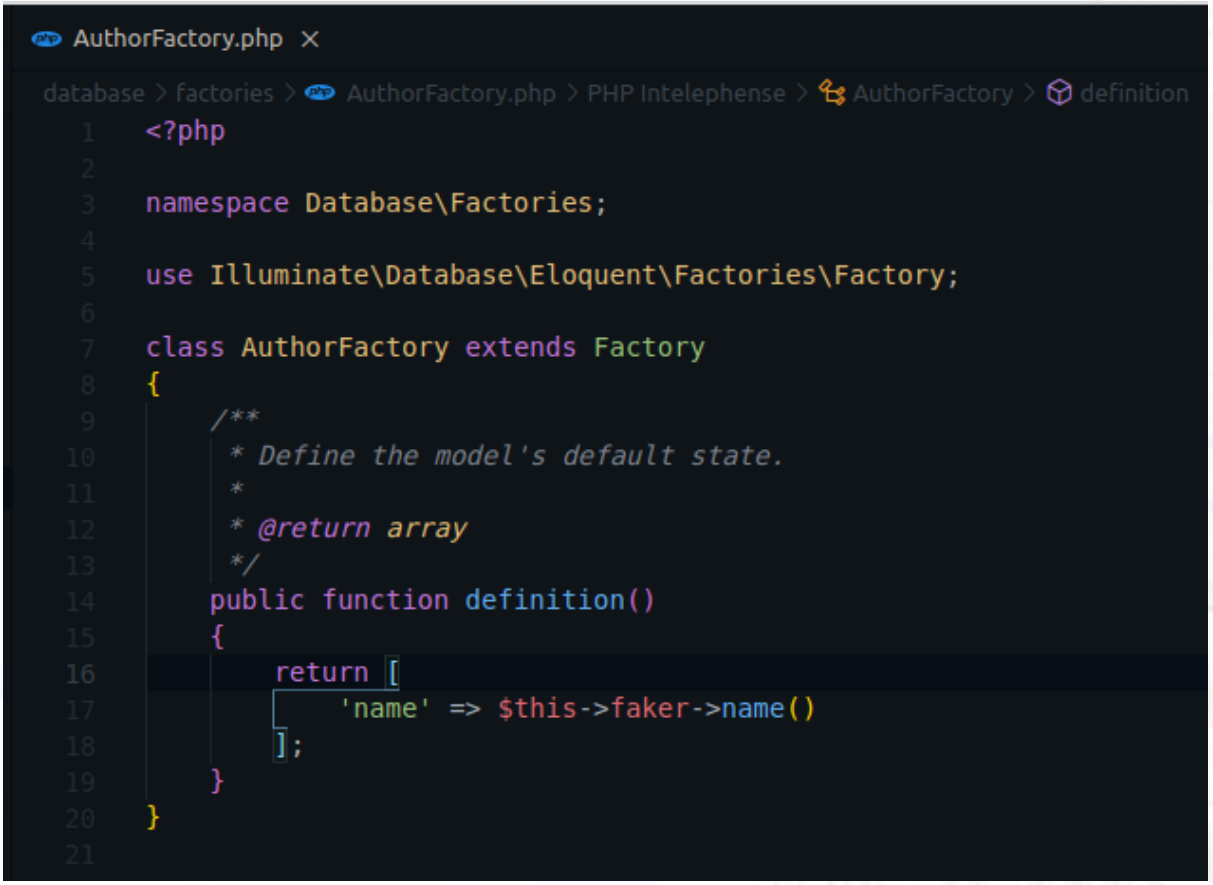
Kemudian kita bisa membuat *file author factory* dan *profile factory* untuk melakukan proses *insert* sample data pada *table* authors dengan cara menjalankan kedua perintah ini

```
php artisan make:factory AuthorFactory
```

Dan menjalankan perintah ini

```
php artisan make:factory ProfileFactory
```

Kemudian ubah isi file database/factories/AuthorFactory.php



Terdapat fungsi `$this->faker->name` yang memiliki fungsi untuk melakukan *generate sample* nama pada *field name*

Untuk detailnya dapat di cek pada dokumentasi *faker*
https://github.com/fzaninotto/Faker

Kemudian kita dapat menjalankan fungsi factory melalui *tinker,* jalankan php artisan tinker dan ikuti step berikut

```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> Author::factory(5)->create();
[!] Aliasing 'Author' to 'App\Models\Author' for this Tinker session.
=> Illuminate\Database\Eloquent\Collection {#3578
     all: [
       App\Models\Author {#3581
         name: "Yvette Medhurst DVM",
         updated_at: "2021-12-23 19:03:38",
         created_at: "2021-12-23 19:03:38",
         id: 1,
       },
       App\Models\Author {#3582
         name: "Dayna Grant MD",
         updated_at: "2021-12-23 19:03:38",
         created_at: "2021-12-23 19:03:38",
         id: 2,
       },
       App\Models\Author {#3583
         name: "Prof. Tanya Mueller",
         updated_at: "2021-12-23 19:03:38",
         created_at: "2021-12-23 19:03:38",
         id: 3,
       },
       App\Models\Author {#3584
         name: "Brad Becker",
         updated_at: "2021-12-23 19:03:38",
         created_at: "2021-12-23 19:03:38",
         id: 4,
       },
       App\Models\Author {#3585
         name: "Kailey Stiedemann",
         updated_at: "2021-12-23 19:03:38",
         created_at: "2021-12-23 19:03:38",
         id: 5,
       },
     ],
   }
```

Kita sudah berhasil membuat 5 sample data author menggunakan factory

Kemudian langkah selanjutnya kita akan melakukan perubahan pada isi *file* database/factories/ProfileFactory.php

Pada ProfileFactory kita sudah menambahkan fungsi `$this->faker->unique()->numberBetween(1, Author::count())` yang digunakan untuk melakukan *set range id* untuk field author_id agar relasi antara *profile* dan *author* dapat terhubung

Kemudian kita akan melakukan *generate sample* data *profile* melalui tinker

```
>>> Profile::factory(5)->create();
[!] Aliasing 'Profile' to 'App\Models\Profile' for this Tinker session.
=> Illuminate\Database\Eloquent\Collection {#3593
     all: [
       App\Models\Profile {#4276
         author_id: 1,
         updated_at: "2021-12-25 07:13:27",
         created_at: "2021-12-25 07:13:27",
         id: 1,
       },
       App\Models\Profile {#3908
         author_id: 3,
         updated_at: "2021-12-25 07:13:27",
         created_at: "2021-12-25 07:13:27",
         id: 2,
       },
       App\Models\Profile {#4527
         author_id: 5,
         updated_at: "2021-12-25 07:13:27",
         created_at: "2021-12-25 07:13:27",
         id: 3,
       },
       App\Models\Profile {#4524
         author_id: 2,
         updated_at: "2021-12-25 07:13:27",
         created_at: "2021-12-25 07:13:27",
         id: 4,
       },
       App\Models\Profile {#4530
         author_id: 4,
         updated_at: "2021-12-25 07:13:27",
         created_at: "2021-12-25 07:13:27",
         id: 5,
       },
     ],
   }
>>>
```

Dapat dilihat pada gambar diatas kita sudah berhasil melakukan *generate sample data* untuk *table profiles*

Sekarang kita dapat melakukan testing melalui tinker, yang pertama kita cek adalah relasi antara *profile* dengan *author* jalankan kembali fungsi tinker dan ikuti step seperti pada gambar dibawah ini

```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> $profile = Profile::find(1);
[!] Aliasing 'Profile' to 'App\Models\Profile' for this Tinker session.
=> App\Models\Profile {#4336
     id: 1,
     author_id: 2,
     created_at: "2021-12-24 06:59:00",
     updated_at: "2021-12-24 06:59:00",
   }
>>> $profile->author->name
=> "Dayna Grant MD"
```

Pada gambar diatas kita telah berhasil menampilkan *field name* yang dimiliki oleh *table authors* melalui *profile*, *method author* merupakan fungsi yang sebelumnya kita set melalui model Profile

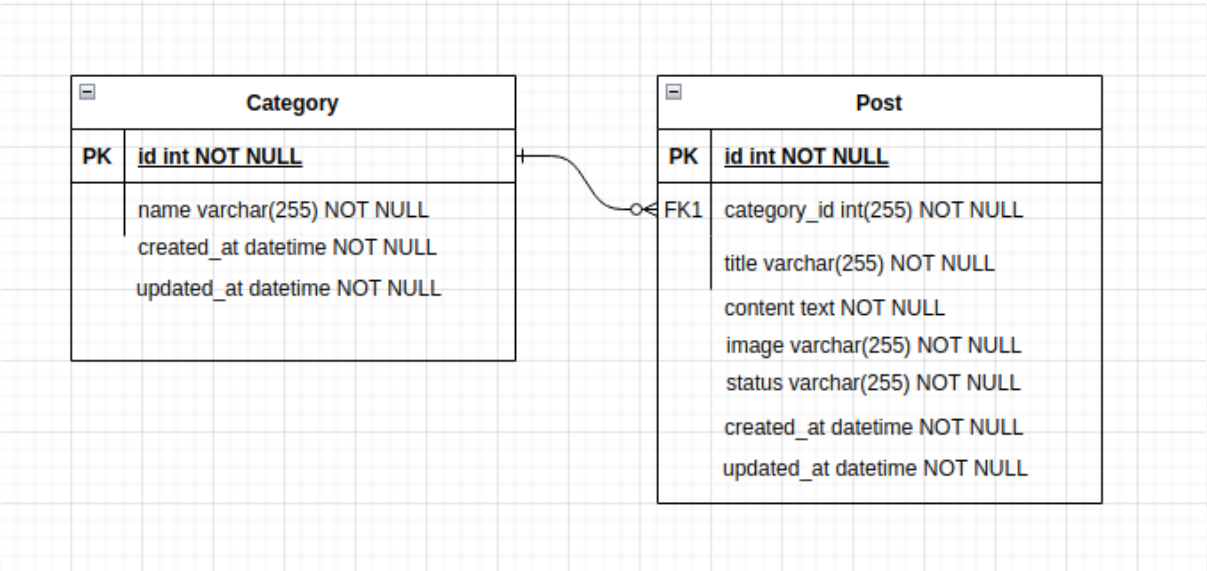
Sekarang kita akan menjalankan sebaliknya yaitu relasi antara *author* ke *profile* melalui tinker

```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> $author = Author::find(1);
[!] Aliasing 'Author' to 'App\Models\Author' for this Tinker session.
=> App\Models\Author {#4336
     id: 1,
     name: "Yvette Medhurst DVM",
     created_at: "2021-12-23 19:03:38",
     updated_at: "2021-12-23 19:03:38",
   }
>>> $author->profile
=> App\Models\Profile {#4479
     id: 3,
     author_id: 1,
     created_at: "2021-12-24 06:59:00",
     updated_at: "2021-12-24 06:59:00",
   }
>>> $author->profile->author_id
=> 1
>>>
```

Dapat kita lihat pada gambar kita sudah berhasil mendapatkan seluruh field yang terdapat pada profile melalui model Author

One to Many

Merupakan relasi dimana 1 *table* memiliki banyak *record data* ke table lainnya disini kita akan mengambil contoh relasi antara model Post dengan Category



Pada contoh diatas kita sudah membuat relasi antara model Category dengan model Post

Selanjutnya kita akan membuat model Post dan model Category, jalankan perintah berikut

```
php artisan make:model Category -m
```

```
php artisan make:model Post -m
```

Kemudian kita akan melakukan perubahan pada *migration table posts*

2021_12_23_063732_create_posts_table.php

database > migrations > 2021_12_23_063732_create_posts_table.php > PHP Intelephense > CreatePostsTable > up > #Function#55ebe600

```php
<?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('content');
            $table->string('status');
            $table->unsignedBigInteger('category_id')->unique();
            $table->foreign('category_id')->references('id')->on('categories');
            $table->timestamps();
        });
    }

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

Pada *migration posts* terdapat *field* category_id yang merupakan *foreign key* dari *id* yang berelasi pada *table categories*, *field* category_id yang berfungsi sebagai penghubung antara relasi *table posts* dan *table categories*

Lakukan perubahan juga pada *migration table categories*

2021_12_25_051755_create_categories_table.php

database > migrations > 2021_12_25_051755_create_categories_table.php > ...

```php
<?php

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

class CreateCategoriesTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

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

Selanjutnya kita juga akan melakukan perubahan pada *file* app/Models/Post.php

Post.php

app > Models > Post.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
    use HasFactory;

    protected $guarded = ['id'];

    public function categories()
    {
        return $this->hasMany(Category::class);
    }
}
```

Pada model Post terdapat *method* dengan nama categories() yang berfungsi untuk membuat relasi dari model Post ke model Category, penamaan jamak merupakan standar penulisan yang disarankan oleh laravel ketika membuat relasi *one to many* pada model

hasMany() merupakan method yang digunakan jika ingin menggunakan relasi dengan model *one to many* yang artinya adalah 1 data *record* pada model Post memiliki banyak *record data* pada model Category

Dan juga kita telah menambahkan property *mass assignment* yang sebelumnya telah kita bahas pada penggunaan relasi one to one

Selanjutnya kita juga akan melakukan perubahan pada *file* app/Models/Category.php

Category.php

app > Models > Category.php > ...

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    use HasFactory;

    protected $guarded = ['id'];

    public function post()
    {
        return $this->belongsTo(Post::class);
    }
}
```

Terdapat method post() yang merupakan relasi antara model *category* ke *post* yang mana method ini juga sebelumnya telah kita bahas pada relasi one to one dimana *belongsTo* mendefinisikan kebalikan dari hubungan *hasMany* menggunakan metode *belongsTo*

Langkah selanjutnya kita akan membuat factory untuk model Post dan model Category
Jalankan perintah berikut untuk membuat kedua *factory*-nya

```
php artisan make:factory PostFactory

php artisan make:factory CategoryFactory
```

Kemudian lakukan perubahan pada *file* database/factories/CategoryFactory.php

CategoryFactory.php

database > factories > CategoryFactory.php > PHP Intelephense > CategoryFactory > definition

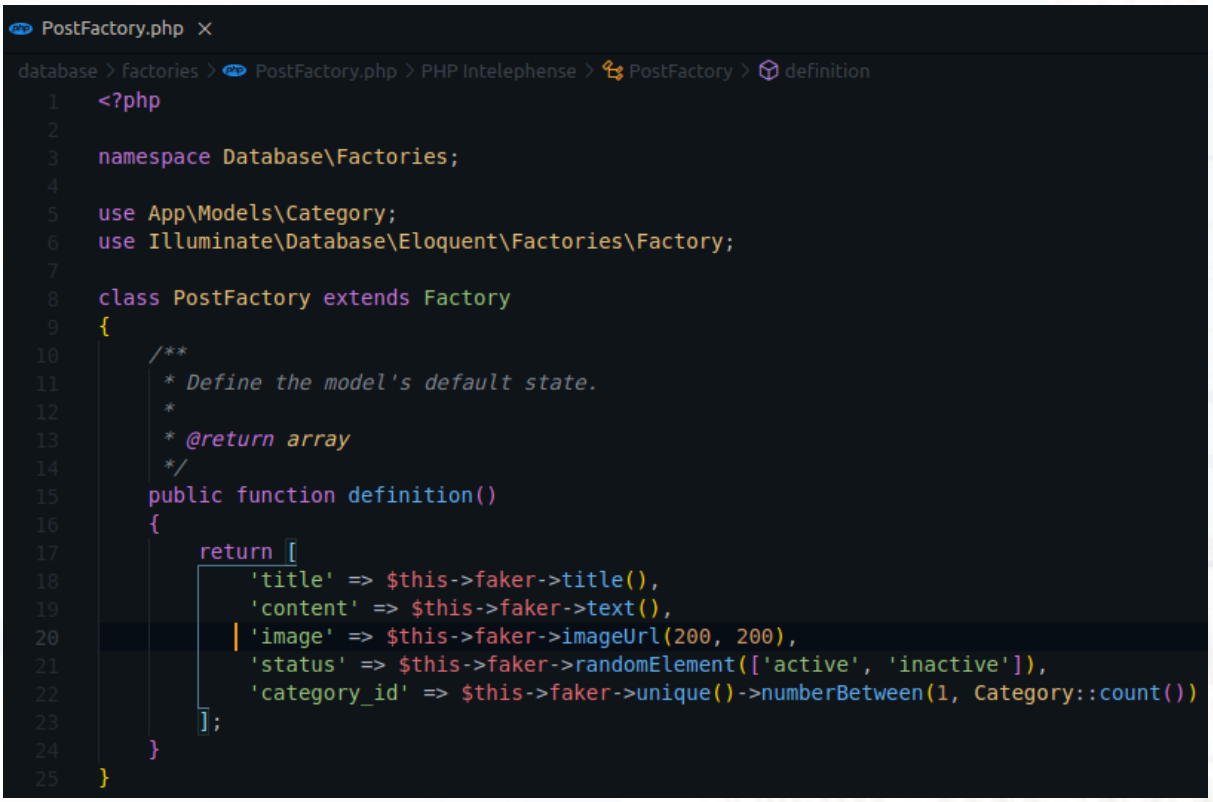
```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

class CategoryFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'name' => $this->faker->text()
        ];
    }
}
```

Pada bagian CategoryFactory terdapat fungsi `$this->faker->text()` yang digunakan untuk mengambil *random text* yang di *insert* ke dalam *field name* pada table *categories*

Kemudian jalankan CategoryFactory melalui tinker untuk *generate 5 sample data* untuk *table categories*

```
>>> Category::factory(5)->create();
[!] Aliasing 'Category' to 'App\Models\Category' for this Tinker session.
=> Illuminate\Database\Eloquent\Collection {#3545
     all: [
       App\Models\Category {#3549
         name: "Suscipit id officia deleniti nesciunt dolores est id. Id
omnis sed adipisci vero. Pariatur ullam dolor consectetur explicabo.",
         updated_at: "2021-12-25 07:19:13",
         created_at: "2021-12-25 07:19:13",
         id: 1,
       },
       App\Models\Category {#3579
         name: "Consectetur ut laboriosam atque magnam corrupti. Possimus
 quia rem in ab nesciunt. In quae sunt in.",
         updated_at: "2021-12-25 07:19:13",
         created_at: "2021-12-25 07:19:13",
         id: 2,
       },
       App\Models\Category {#3547
         name: "Dolor veniam non aut laborum. Velit itaque debitis error
rerum culpa ad commodi. Fuga sit sint ea ea eos occaecati qui.",
         updated_at: "2021-12-25 07:19:13",
         created_at: "2021-12-25 07:19:13",
         id: 3,
       },
       App\Models\Category {#4380
         name: "Qui voluptas voluptate molestiae quia nihil minima volupt
atem. Itaque quia accusantium soluta assumenda voluptate. Omnis unde ea n
am ea harum distinctio.",
         updated_at: "2021-12-25 07:19:13",
         created_at: "2021-12-25 07:19:13",
         id: 4,
       },
```

Kemudian kita lakukan perubahan pada isi *file* database/factories/PostFactory.php



Pada PostFactory terdapat fungsi `$this->faker->imageUrl(200, 200)` yang digunakan untuk membuat *sample data image url* dengan ukuran dimensi tinggi 200 dan lebar 200 serta terdapat fungsi `$this->faker->randomElement(['active', 'inactive'])` yang digunakan untuk melakukan *set flag status* secara *random*

Kemudian kita bisa lakukan proses *generate* 5 *sample data* PostFactory melalui *tinker*

```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> Post::factory(5)->create();
[!] Aliasing 'Post' to 'App\Models\Post' for this Tinker session.
=> Illuminate\Database\Eloquent\Collection {#3578
     all: [
       App\Models\Post {#3589
         title: "Dr.",
         content: "Fugiat ducimus ea in eum. Minima ab placeat quaerat qu
i qui. Fugiat provident at et corrupti iste architecto. Quis deleniti rep
ellendus dolorem.",
         image: "https://via.placeholder.com/200x200.png/0088dd?text=nemo
",
         status: "inactive",
         category_id: 2,
         updated_at: "2021-12-25 07:24:53",
         created_at: "2021-12-25 07:24:53",
         id: 1,
       },
       App\Models\Post {#3588
         title: "Mrs.",
         content: "Nulla est voluptatibus impedit quae. Dolores dicta duc
imus molestiae et ut hic. Adipisci voluptate mollitia eligendi. Perspicia
tis error aut dolorem velit ipsam error minima.",
         image: "https://via.placeholder.com/200x200.png/00bb33?text=at",
         status: "active",
         category_id: 1,
         updated_at: "2021-12-25 07:24:53",
         created_at: "2021-12-25 07:24:53",
         id: 2,
       },
       App\Models\Post {#4375
         title: "Dr.",
         content: "Quaerat vel et blanditiis labore. Quia accusamus culpa
 fugit voluptate enim veniam adipisci. Voluptate ratione consectetur moll
itia nam optio. Consequuntur in explicabo qui ut delectus.",
         image: "https://via.placeholder.com/200x200.png/00ee22?text=qui"
,
         status: "inactive",
         category_id: 3,
         updated_at: "2021-12-25 07:24:53",
```

Kita sudah berhasil melakukan *generate* pada kedua *table categories* dan *posts* langkah selanjutnya kita bisa melakukan uji coba relasi *one to many* melalui *tinker*, pertama kita akan melakukan *testing* pada relasi *table posts* ke *table categories*

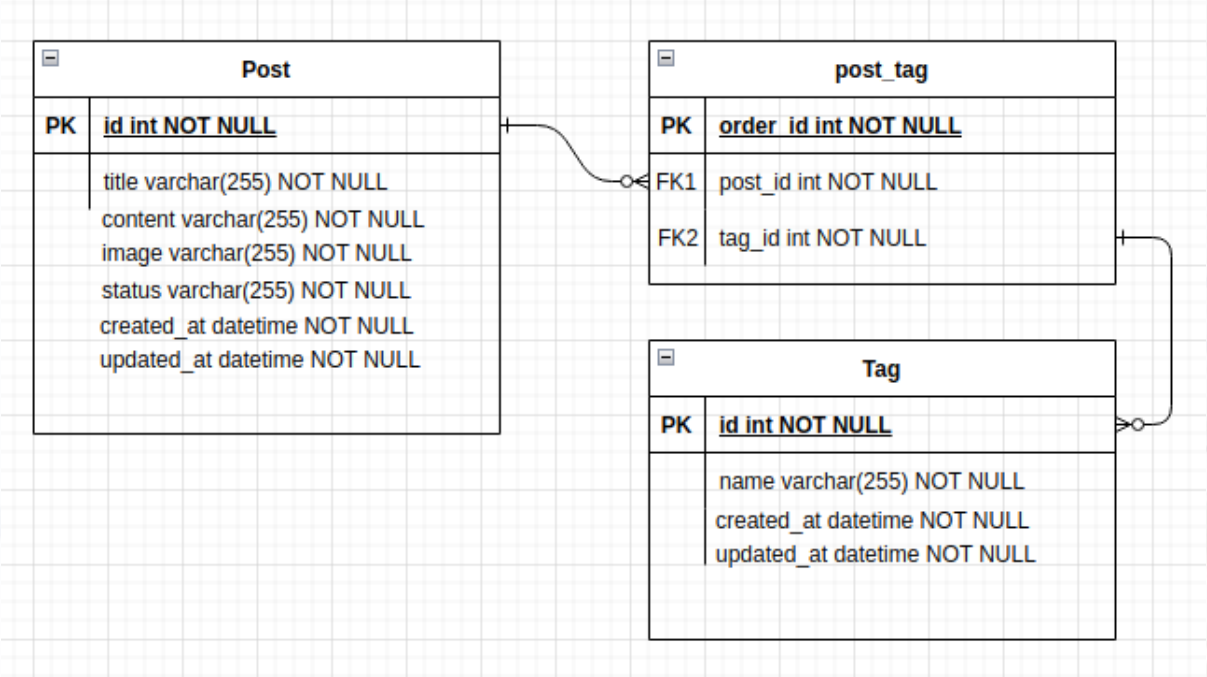
```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> $post = Post::find(1);
[!] Aliasing 'Post' to 'App\Models\Post' for this Tinker session.
=> App\Models\Post {#4336
     id: 1,
     title: "Dr.",
     content: "Fugiat ducimus ea in eum. Minima ab placeat quaerat qui qu
i. Fugiat provident at et corrupti iste architecto. Quis deleniti repelle
ndus dolorem.",
     image: "https://via.placeholder.com/200x200.png/0088dd?text=nemo",
     status: "inactive",
     created_at: "2021-12-25 07:24:53",
     updated_at: "2021-12-25 07:24:53",
     category_id: 2,
   }
>>> $post->category
=> App\Models\Category {#4232
     id: 2,
     name: "Consectetur ut laboriosam atque magnam corrupti. Possimus qui
a rem in ab nesciunt. In quae sunt in.",
     created_at: "2021-12-25 07:19:13",
     updated_at: "2021-12-25 07:19:13",
   }
>>> $post->category->name
=> "Consectetur ut laboriosam atque magnam corrupti. Possimus quia rem in
 ab nesciunt. In quae sunt in."
>>>
```

Kemudian kita bisa lakukan pengecekan relasi dari *table categories* ke *table posts*

```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> $category = Category::find(1);
[!] Aliasing 'Category' to 'App\Models\Category' for this Tinker session.
=> App\Models\Category {#4336
     id: 1,
     name: "Suscipit id officia deleniti nesciunt dolores est id. Id omni
s sed adipisci vero. Pariatur ullam dolor consectetur explicabo.",
     created_at: "2021-12-25 07:19:13",
     updated_at: "2021-12-25 07:19:13",
   }
>>> $category->posts
=> Illuminate\Database\Eloquent\Collection {#4271
     all: [
       App\Models\Post {#4479
         id: 2,
         title: "Mrs.",
         content: "Nulla est voluptatibus impedit quae. Dolores dicta duc
imus molestiae et ut hic. Adipisci voluptate mollitia eligendi. Perspicia
tis error aut dolorem velit ipsam error minima.",
         image: "https://via.placeholder.com/200x200.png/00bb33?text=at",
         status: "active",
         created_at: "2021-12-25 07:24:53",
         updated_at: "2021-12-25 07:24:53",
         category_id: 1,
       },
     ],
   }
>>>
```

Many to Many

Pada relasi *many to many* terdapat banyak relasi ke relasi *table* yang lain pada pembahasan kali ini kita akan menggunakan *table posts* dan *tags* untuk menerapkan relasi *many to many*



Pertama kita perlu membuat model Tag jalankan perintah berikut

```
php artisan make:model Tag -m
```

Kemudian kita bisa melakukan perubahan pada isi *file* app/Models/Tag.php

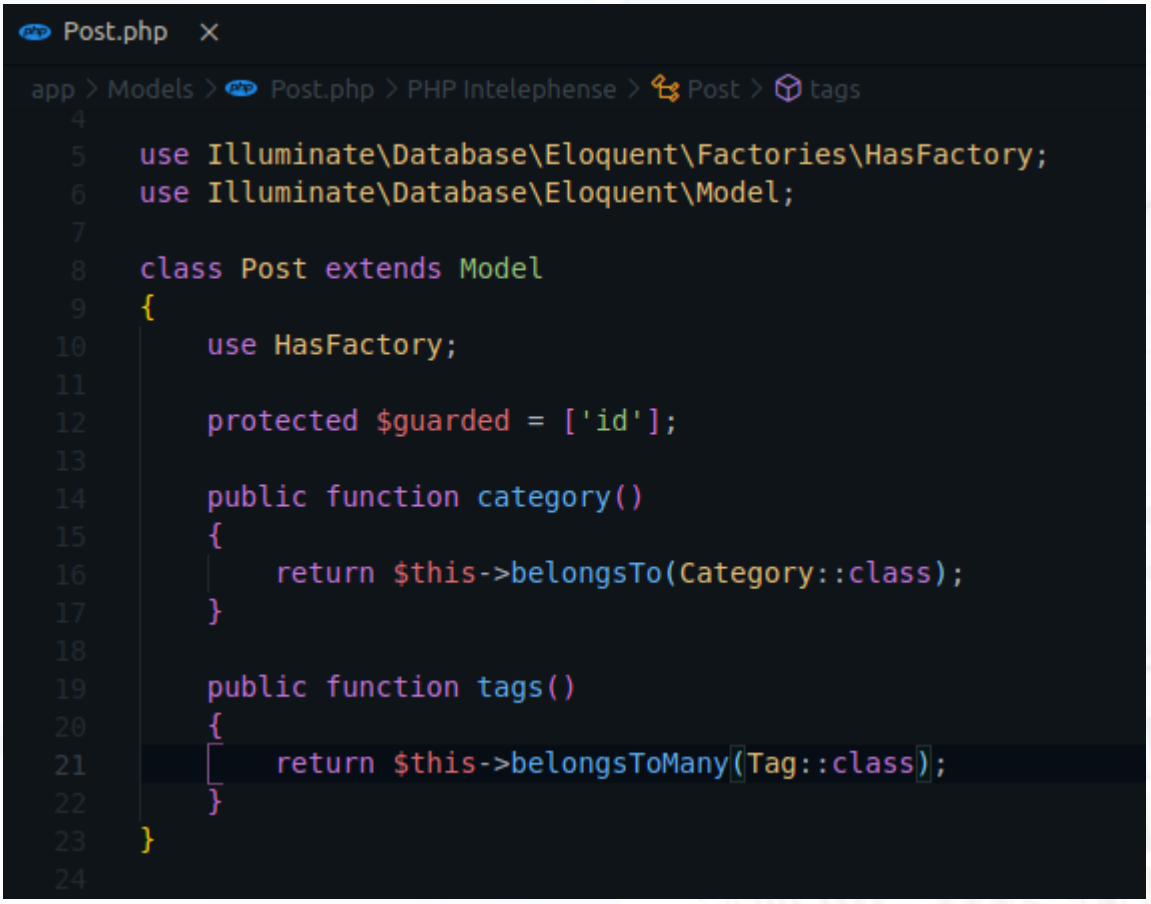
Tag.php

app > Models > Tag.php > PHP Intelephense > Tag > posts

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Tag extends Model
{
    use HasFactory;

    protected $guarded = ['id'];

    public function posts()
    {
        return $this->belongsToMany(Post::class);
    }
}
```

Terdapat *method posts()* yang digunakan untuk membuat relasi antara *table tags* ke *table posts*, penamaan *method* menggunakan kata jamak yang merupakan konvensi yang dianjurkan oleh laravel ketika *table* tersebut memiliki relasi yang banyak *(many)*

Kemudian kita bisa melakukan perubahan pada isi *file* app/Models/Post.php



Pada model Post terdapat penambahan method *tags()* yang berfungsi sebagai relasi antara *table posts* dengan *table tags* dan terdapat fungsi *belongsToMany* yang merupakan *method* yang digunakan untuk relasi *many to many*

Kemudian langkah selanjutnya kita dapat membuat TagFactory untuk membuat *sample* data pada model Tag

Jalankan perintah berikut

```
php artisan make:factory TagFactory
```

Kemudian ubah isi file database/factories/TagFactory.php

TagFactory.php

database > factories > TagFactory.php > PHP Intelephense > TagFactory > definition

```php
<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

class TagFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array
     */
    public function definition()
    {
        return [
            'name' => $this->faker->sentence()
        ];
    }
}

```

Terdapat fungsi $this->faker->sentence() yang digunakan untuk membuat teks *dummy* pada *field name* di *table tags*

Kemudian langkah selanjutnya kita bisa lakukan *generate sample* data pada *table tags* menggunakan tinker

```
temmy@temmy-virtuoso:~/investree-dev$ php artisan tinker
Psy Shell v0.10.12 (PHP 7.4.27 — cli) by Justin Hileman
>>> Tag::factory(5)->create();
[!] Aliasing 'Tag' to 'App\Models\Tag' for this Tinker session.
=> Illuminate\Database\Eloquent\Collection {#3578
     all: [
       App\Models\Tag {#3581
         name: "Molestias ut facilis vitae pariatur.",
         updated_at: "2021-12-25 12:49:25",
         created_at: "2021-12-25 12:49:25",
         id: 1,
       },
       App\Models\Tag {#3582
         name: "Quisquam eos ullam voluptate nostrum iste voluptatem.",
         updated_at: "2021-12-25 12:49:25",
         created_at: "2021-12-25 12:49:25",
         id: 2,
       },
       App\Models\Tag {#3583
         name: "Tempore sed modi accusantium voluptatem.",
         updated_at: "2021-12-25 12:49:25",
         created_at: "2021-12-25 12:49:25",
         id: 3,
       },
       App\Models\Tag {#3584
         name: "Sunt incidunt sed molestias.",
         updated_at: "2021-12-25 12:49:25",
         created_at: "2021-12-25 12:49:25",
         id: 4,
       },
       App\Models\Tag {#3585
         name: "Quam quaerat consequuntur nihil eveniet quo.",
         updated_at: "2021-12-25 12:49:25",
         created_at: "2021-12-25 12:49:25",
         id: 5,
       },
     ],
```

Kemudian kita bisa lakukan *insert data* pada *pivot table* melalui *file* database/seeders/DatabaseSeeder.php

Pada bagian data DatabaseSeeder kita melakukan pemanggilan seluruh data pada model Tag

```
$tags = Tag::all();
```

Kemudian kita lakukan *looping* pada seluruh data dari *table posts* dan terdapat pemanggilan relasi dari *table posts* ke *table tags* dengan menggunakan *method attach* proses *insert* ke dalam *table* pivot dilakukan

```
Post::all()->each(function ($post) use ($tags) {
      $post->tags()->attach(
            $tags->random(rand(1,5))->pluck('id')->toArray()
      );
});
```

pluck() merupakan salah satu method *collection* yang digunakan untuk mengambil salah satu *field* pada data berbentuk *collection*

Untuk detailnya dapat dipelajari
https://laravel.com/docs/8.x/collections#method-pluck

Method run() digunakan apabila kita menjalankan perintah menggunakan

```
php artisan db:seed
```

maka proses *insert* data *dummy* dilakukan pada laravel juga kita dapat menjalankan *seeder per class*

Bisa di cek melalui dokumentasi nya https://laravel.com/docs/8.x/seeding

Kemudian kita melakukan uji coba untuk mengecek relasi *many to many* yang sudah dibuat melalui tinker, kita akan mencoba relasi dari *table post* ke *table tags*

```
>>> $post = Post::find(1);
=> App\Models\Post {#3541
     id: 1,
     title: "Miss",
     content: "Quaerat ut omnis sapiente animi. Dolore molestias conse
quatur necessitatibus totam in. Eum quae vitae enim non est.",
     image: "https://via.placeholder.com/200x200.png/00ccee?text=rerum
",
     status: "active",
     created_at: "2021-12-27 17:50:33",
     updated_at: "2021-12-27 17:50:33",
     category_id: 4,
   }
>>> $post->tags;
=> Illuminate\Database\Eloquent\Collection {#4486
     all: [
       App\Models\Tag {#4485
         id: 4,
         name: "Sunt incidunt sed molestias.",
         created_at: "2021-12-25 12:49:25",
         updated_at: "2021-12-25 12:49:25",
         pivot: Illuminate\Database\Eloquent\Relations\Pivot {#3864
           post_id: 1,
           tag_id: 4,
         },
       },
     ],
   }
```

Dapat dilihat pada gambar diatas relasi dari *table posts* ke *table tags* sudah berhasil

Kemudian kita bisa mencoba sebaliknya dari *table tags* ke *table posts*

```
>>> $tag = Tag::find(1);
[!] Aliasing 'Tag' to 'App\Models\Tag' for this Tinker session.
=> App\Models\Tag {#4483
     id: 1,
     name: "Molestias ut facilis vitae pariatur.",
     created_at: "2021-12-25 12:49:25",
     updated_at: "2021-12-25 12:49:25",
   }
>>> $tag->posts
=> Illuminate\Database\Eloquent\Collection {#4493
     all: [
       App\Models\Post {#4494
         id: 2,
         title: "Miss",
         content: "Laudantium nostrum quos sunt error. Nisi neque dict
a et explicabo. Impedit animi qui ipsum dolores est adipisci corporis
similique. Illum voluptates consequuntur eos sunt quo repudiandae odit
.",
         image: "https://via.placeholder.com/200x200.png/008844?text=d
olor",
         status: "inactive",
         created_at: "2021-12-27 17:50:33",
         updated_at: "2021-12-27 17:50:33",
         category_id: 1,
         pivot: Illuminate\Database\Eloquent\Relations\Pivot {#4497
           tag_id: 1,
           post_id: 2,
         },
       },
```

Maka dapat dilihat pada gambar diatas relasi *many to many* dari *table tags* ke *posts* sudah berhasil