**Eloquent Relationship Documentation**

**--:Index:--**

03 Many To Many relationship \_ belongsToMany() \_ Laravel Eloquent Relationships

4.1) has-one-through - has-many-through

Exm:

|  |  |  |
| --- | --- | --- |
| * + Project   + 1) id   + 2) title | * + User   + 1) id   + 2) project\_id | * + Task   + 1) id   + 2) user\_id |

Here< A project has many User and A user Has many Task. I have to create a relationship A project has many Task using has-on-through & has-may-through realationship

\*\*) Create Relationship In Project Model

classProjectextendsModel

{

    useHasFactory;

// basic relationship

    publicfunctionusers(){

        return$this->hasMany(User::class);

    }

// vvirelatinshiop

    publicfunctiontasks(){

        // return $this->hasOneThrough(Task::class, User::class);

        return$this->hasManyThrough(Task::class, User::class,'project\_id','user\_id');

    }

}

\*\*) create relationship in User model:

classUserextendsAuthenticatable

{

    useHasApiTokens, HasFactory, Notifiable;

    // #### has on through relation ship

    publicfunctionproject(){

        return$this->belongsTo(Project::class);

    }

publicfunctiontasks(){

        return$this->hasMany(Task::class);

    }

}

\*\*) create relationship in Task model:

classTaskextendsModel

{

    useHasFactory;

    protected$guarded= [];

    publicfunctionuser(){

        return$this->belongsTo(User::class);

    }

}

\*\*) controller code:

Route::get('/project-has-many-through',function(){

    $project=Project::with(['tasks'])->first();

    return$project;

});

4.2) has many relation(belongsToMany) with pivot table && ship has-many-through with pivot table

projects

    id

users

    id

project\_user

    project\_id

    user\_id

tasks

    id

    user\_id

Here, Project has many user and User table has many project that’s why we make to a pivot table name in project\_user. A user has many task. I want to create relation ship is A project has many task, that is A project relation with tasks table “hasManyThrough” of project\_user pivot table

1=>) make model ProjectOne

classProjectOneextendsModel

{

    useHasFactory;

    protected$guarded= [];

    publicfunctionusers(){

        return  $this->belongsToMany(UserOne::class, 'project\_user\_one', 'project\_id' ,'user\_id');

    }

    publicfunctiontasks(){

        return$this->hasManyThrough(

            TaskOne::class,

            ProjectoneUserOnePivote::class,

            'project\_id', // project\_id is foreign key of pivot table

            'user\_id', // user\_id for taskOne table

            'id', // id is ProjectOne table,

            'user\_id'// user\_id for pivot table

        );

    }

}

Factory &&Seeder Creae:

Php artisan make:factoryProjectOneFactory –model= ProjectOne

Php artisan make:seederProjectOneSeeder

2=) UserOne model create:--

classUserOneextendsModel

{

    useHasFactory;

    protected$guarded= [];

    publicfunctionprojects(){

        return$this->belongsToMany(ProjectOne::class,'project\_user\_one','user\_id','project\_id');

    }

    publicfunctiontasks(){

        return$this->hasManyThrough(

            TaskOne::class,

            ProjectoneUserOnePivote::class,

            'project\_id', // project\_id is foreign key of pivot table

            'user\_id', // user\_id for taskOne table

            'id', // id is UserOne table,

            'user\_id'// user\_id for pivot table

        );

    }

}

* Same as UserOneFactory and UserOneSeeder Create

3=) ProjectoneUserOnePivote model create and ProjectoneUserOnePivoteFactory and seeder create

4=>) TaskOne model create and TaskOneFactoryand seeder create

**5.1 One to Many - One to One Polymorphic Relationships Laravel Eloquent Relationships**

User

post

video

broadcust

event

comment

user\_id

    commentable\_id   //1

    commentable\_type  //post

Here, One USER can comment post, video, broadcast and event too. In this situation you can handel this using one2One or one2many polymorphic relationship

Exp::-

Post Model:-

classPostextendsModel

{

    useHasFactory;

    protected$guarded=[];

    publicfunctionuser(){

        return$this->belongsTo(User::class)->withDefault([

            'name'=>"Guest User"

        ]);

    }

    publicfunctiontags(){

        return$this->belongsToMany(Tag::class,'post\_tag','post\_id','tag\_id')

        ->withTimestamps()

        ->withPivot('status');

    }

    // ###5.1 One to Many - One to One Polymorphic Relationships Laravel Eloquent Relationships

    publicfunctioncomments(){

        return$this->morphMany(Comment::class,'commentable');

    }

    publicfunctioncomment(){

        return$this->morphOne(Comment::class,'commentable');

    }

}

Video model:-

classVideoextendsModel

{

    useHasFactory;

    protected$fillable= [

        'title'

    ];

     // ###5.1 One to Many - One to One Polymorphic Relationships Laravel Eloquent Relationships

     publicfunctioncomments(){

        return$this->morphMany(Comment::class,'commentable');

    }

}

Comment model:-

classCommentextendsModel

{

    useHasFactory;

    protected$fillable= [

        'body',

        'user\_id',

        'commentable\_id',

        'commentable\_type',

    ];

// ###5.1 One to Many - One to One Polymorphic Relationships Laravel Eloquent Relationships

    publicfunctioncommentable()

    {

        return$this->morphTo();

    }

}

## controller code:-

// 5.1 One to Many - One to One Polymorphic Relationships Laravel Eloquent Relationships

Route::get('/one2one-one2many-polymorfic',function(){

    // dd( User::all()->random()->id);

    // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* One2One & one2Many Polymorfic relationship create data \*\*\*\*

    $userid=  User::find(30)->id;

    $post=Post::create([

    'user\_id'=>$userid,

    'title'=>"This is first title"

    ]);

    $video=Video::create([

        'title'=>"This is Video title"

    ]);

    // one2Many polymorfic relationship create

    $post->comments()->create([

        'user\_id'=>$userid,

        'body'=>'This is text body'

    ]);

    $video->comments()->create([

        'user\_id'=>$userid,

        'body'=>'This is Video body'

    ]);

   return"One to many polymorfic relationship create successfully";

//    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* One2One & one2Many Polymorfic relationship create data \*\*\*\*

\*\*\*\*findPostComent\*\*\*

    $post=Post::find(97)->load('comments');

    dd($post);

    \*\*\*\*findcommenttomodel\*\*\*

    $coment=Comment::find(1)->load('commentable');

    dd($coment );

});

**5.2 Many to Many Polymorphic Relationship Laravel Eloquent Relationships.mp4**

🡺php artisan make:migrationcreate\_taggables\_table --create=taggables