## [Defining Relationships](https://laravel.com/docs/5.7/eloquent-relationships" \l "defining-relationships)

### One To One

A one-to-one relationship is a very basic relation. For example, a User model might be associated with one Phone. To define this relationship, we place a phone method on the Usermodel. The phone method should call the hasOne method and return its result:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class User extends Model{

/\*\*

\* Get the phone record associated with the user.

\*/

public function phone()

{

return $this->hasOne('App\Phone');

}}

The first argument passed to the hasOne method is the name of the related model. Once the relationship is defined, we may retrieve the related record using Eloquent's dynamic properties. Dynamic properties allow you to access relationship methods as if they were properties defined on the model:

#### **Defining The Inverse Of The Relationship**

So, we can access the Phone model from our User. Now, let's define a relationship on the Phone model that will let us access the User that owns the phone. We can define the inverse of a hasOne relationship using the belongsTo method:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Phone extends Model{

/\*\*

\* Get the user that owns the phone.

\*/

public function user()

{

return $this->belongsTo('App\User');

}}

/

### One To Many

A "one-to-many" relationship is used to define relationships where a single model owns any amount of other models. For example, a blog post may have an infinite number of comments. Like all other Eloquent relationships, one-to-many relationships are defined by placing a function on your Eloquent model:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model{

/\*\*

\* Get the comments for the blog post.

\*/

public function comments()

{

return $this->hasMany('App\Comment');

}}

### One To Many (Inverse)

Now that we can access all of a post's comments, let's define a relationship to allow a comment to access its parent post. To define the inverse of a hasMany relationship, define a relationship function on the child model which calls the belongsTo method:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Comment extends Model{

/\*\*

\* Get the post that owns the comment.

\*/

public function post()

{

return $this->belongsTo('App\Post');

}}

### Many To Many

Many-to-many relations are slightly more complicated than hasOne and hasManyrelationships. An example of such a relationship is a user with many roles, where the roles are also shared by other users. For example, many users may have the role of "Admin". To define this relationship, three database tables are needed: users, roles, and role\_user. The role\_user table is derived from the alphabetical order of the related model names, and contains the user\_id and role\_id columns.