### 数据迁移

(1)<https://laravel-china.org/docs/laravel/5.5/> 官方文档

php artisan make:model name;//创建数据映射Bean

php artisan make:migration create\_users\_table --create=users

php artisan make:migration add\_column\_name\_into\_users --table=users

protected $fillable = ['name', 'age', ];//设置字段可填充

1. 解决key文件丢失:php artisan passport:install

### 数据的CURD

·$table()->String(‘name’)->first();//将该字段放到第一位

·$table()->String(‘name’)->after(‘id’);

·DB::table(name)->insertGetId([a=>a,b=>b]);//添加一条数据并获得id

·truncate() >>>>> delete();//删除某个表的数据

·DB::table(name)->where(‘id’,2)->update([关联数组]);

·increment();decrement();//针对表中的所有指定字段

·DB::table(name)->where()->orWhere();

· $u=DB::table('users')->whereIn('id',[1,5,6])->orWhereNotIn('id',[5])->get()

### 3.数据的映射

·Laravel ORM映射的时候,默认使用domin\_id作为外键关联,若不是的话可以进行覆盖

·**OneToOne**

· hasOne();

· belongsTo();关联允许定义默认模型，这适应于当关联结果返回的是 null的情况

$this->belongsTo('App\User')->withDefault([ 'name' => '游客',]);

·**OneToMany**

·hasMany（);

·belongsTo（);

·

·**ManyToMany**

·多对多的关系需要一张中间表来维护关系（users,roles,user\_role）

·$this->belongsToMany();接收四个可选参数

1.domin的命名空间(包)

2.中间表名

3.该domin对应在中间表的列名

4.该domin对应的关系在中间表对应的列名

·需要注意的是，我们取得的每个 Role 模型对象，都会被自动赋予pivot 属性,它代表中间表的一个模型对象，能像其它的 Eloquent模型一样使用。

·

**·远层一对多**

return $this->hasManyThrough(

'App\Post',

'App\User',

'country\_id', // 用户表外键...

'user\_id', // 文章表外键...

'id', // 国家表本地键...

'id' // 用户表本地键...

);

**·关联数据计数**

·$r=App\User::where('id',1)->withCount(

'dogs'=>function($query){

$query->where('age','>',2);

}

)->get();

**·关联数据查询过滤**

·has();

·whereHas();

·doesntHave();

**·关联数据查询预加载**

·解决问题 （查询N+1的问题）

$books = App\Book::all();

foreach ($books as $book) {

echo $book->author->name;

}

·$books = App\Book::with('author')->get();

·预加载多个关联

$books = App\Book::with(['author','author'])->get();

·还可以给预加载添加约束条件

**·插入&更新关联模型**

·save()方法

$dog=new App\Dog(['name'=>'asdas','age'=>1]);

$u=App\User::find(1);

$u->dogs()->save($dog);

·create()方法

$dog=$u->dogs()->create([

‘name’=>’ddd’,

‘age’=>12

]);

·您可以使用createMany（）方法保存多个关联模型

$u->dogs()->createMany([

[

‘name’=>’ddd’,

‘age’=>12

],

[

‘name’=>’ddd’,

‘age’=>12

]

]);

·保存关联类型模型 只能由一方 save（），create（），createMany（）来保存所关联的多方数据模型

**·更新belongsTo关联**

·当更新 belongsTo 关联时，可以使用associate方法。此方法会在子模型中设置外键：

**·多对多关联**

·$user->roles()->attach($roleId);//往中间表添加多对多关联

·$user->roles()->detach($roleId);//移除多对多的关联

**·同步关联**

· 您也可以使用sync方法来构造多对多关联。sync方法可以接收 ID 数组，向中间表插入对应关联数据记录。所有没放在数组里的 IDs 都会从中间表里移除。

$user->roles()->sync([1, 2, 3]);

·您可以通过 ID 传递其他额外的数据到中间表：

$user->roles()->sync([1 => ['expires' => true], 2, 3]);

·如果您不想移除现有的 IDs，可以使用 syncWithoutDetaching 方法：

$user->roles()->syncWithoutDetaching([1, 2, 3]);

·protected $touches=[‘prent’];//子模型更新的时候自动更新父模型的update\_at

### Eloquent

·protected $table = 'my\_flights';//映射哪张Table

·Eloquent 也会假定每个数据表都有一个名为id的主键字段。你可以定义一个受保护的$primaryKey属性来覆盖这个约定。

·在模型内设置 $dateFormat 属性。这个属性决定了日期属性应如何存储在数据库中，以及模型被序列化成数组或 JSON 时的格式：

protected $dateFormat = 'U';

· const CREATED\_AT = 'creation\_date';

const UPDATED\_AT = 'last\_update';

·cursor 允许你使用游标来遍历数据库数据，该游标只执行一个查询。处理大量数据时，可以使用 cursor 方法可以大幅度减少内存的使用量：

foreach (Flight::where('foo', 'bar')->cursor() as $flight) {

//

}

$flights = App\Flight::find([1, 2, 3]);

·如果你希望在找不到模型时抛出异常，可以使用 findOrFail 以及 firstOrFail 方法。这些方法会检索查询的第一个结果。如果没有找到相应结果，就会抛出一个 Illuminate\Database\Eloquent\ModelNotFoundException：

如果没有对异常进行捕获，则会自动返回 HTTP 404 响应给用户。

**·模型的创建方法**

·save();

#### ·firstOrCreate()/ firstOrNew();

#### ·updateOrCreate

### 数据库的分页

·desc 降序 esc为升序

·php 中数组的截取 array\_slice()

### 日期的转换

·{{date('Y-m-d',strtotime($new\_linetime))}} //日期的转换（前端）

·

### 页面的依赖注入

@inject('timelines', 'App\Repositories\NoticeRepository')

之后可以在本页面中调用该仓库中的任何方法进行数据处理。

### Send-clound邮箱的使用

·composer require naux/sendcloud

·'providers' => [ // 添加这行 Naux\Mail\SendCloudServiceProvider::class,

];

### Laravel 的自定义验证规则

(1)在AppServiceProvider 的boot方法中加入:

Validator::extend('code',function ($attribute,$value,$p){

return $value==session('tel\_code');

});

(2)在validation中加入自定义的名称以及提示信息

(3)在控制器的validator中添加相应的规则名称

**13.collection集合常用方法**

(1)$collection->flatten(1);取出集合中指定深度的数据

(2)$collection->filter();接受一个闭包函数返回集合中符合条件的元素,如果没有提供回调函数，集合中所有返回 false 的元素都会被移除.

(3)forget()来移除集合中对应的内容

(4)$collection->groupBy('account\_id');根据集合中的指定key进行分组

(5)$collection->has('product');判断集合中是否存在指定的key