20.模型的作用域

学习要点：

1.本地作用域

2.全局作用域

本节课我们来开始学习数据库模型的本地作用域和全局作用域的设置方法。

一. 本地作用域

1. 很多情况下，我们在数据查找时有一部分条件会被重复且大量使用；
2. 而这个条件，可能只是在这个模型对应的数据表使用，别的表并不使用；
3. 那么这种情况，可以使用本地作用域的方式，将常用的 SQL 封装起来；
4. 比如：用户模块中，我们大量查询需要查询性别为男，且其它条件的 SQL；

```
$users = User::where('gender', '男')
            ->where('price', '>', 90)
            ->get();
```

PS：我们可以将性别为男这个片段，封装成一个单独的方法，然后统一在这个模型下调用；

```
//App\Http\Models;
//本地作用域，搜索自动添加为“男”的条件
//语法：scope 开头，后面名称尽可能包含语义
public function scopeGenderMale($query)
{
    return $query->where('gender', '男');
}

//当然，如果赶紧单词太长，直接 gm()也行
$users = User::genderMale()
            ->where('price', '>', 90)
            ->get();
```

5. 上面的方法比较死板，适合简单粗暴，如果想要灵活多变，支持传递参数；

```
//参数可以是 1 个或多个
$users = User::gender('女', -3)
            ->where('price', '>', 90)
            ->get();

//参数 2 和 3，接受控制器传递过来的 1，2
public function scopeGender($query, $value, $value2)
{
    return $query->where('gender', $value)->where('status', $value2);
}
```

二．全局作用域

1. 全局作用域，顾名思义就是在任意地方都可以有效的封装条件；
2. 比如有个需求，不管在哪里操作，总是显示 status 为 1 的用户；
3. 首先在 app 目录下创建一个用于全局作用域的目录：Scopes；
4. 创建一个用于设置 status 为 1 的全局作用域的类，它需要实现 scope 接口；

```php
namespace App\Scopes;

//这里引用代码自动生成
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Scope;

class StatusScope implements Scope
{

    /**
     * @inheritDoc
     */
    public function apply(Builder $builder, Model $model)
    {
        // TODO: Implement apply() method.
        return $builder->where('status', 1);
    }
}
```

5. 此时，还不能实现全局，因为需要在模型设置个开关，让其富有灵活性；

```php
//启用全局作用域
protected static function booted()
{
    parent::booted(); // TODO: Change the autogenerated stub
    static::addGlobalScope(new StatusScope());
}
```

PS：而在控制器端，并不需要做任何设置，即可自动添加 status=1 的条件；

6. 当然，如果这个全局只是针对某个模块，并不需要创建一个全局类，直接闭包即可；

```php
static::addGlobalScope('status', function (Builder $builder) {
    return $builder->where('status', 1);
});
```

PS：注意 Builder 引入的文件和全局类引入的文件一致，如果引入别的同名类会错；

7. 如果某个查询，并不需要这个全局条件，可以单独移出掉；

```php
//取消名称为 status 的全局
$users = User::withoutGlobalScope('status')->get();
//取消全局类的条件
$users = User::withoutGlobalScope(StatusScope::class)->get();
```

PS：还有 withoutGlobalScopes([])方法，传递参数取消多个全局；