

基于 laravel5.7开发博客笔记

日期时间处理包 Carbon

安装

由于 Laravel 框架已经默认安装了 Carbon 包，所以不用安装。如果其他项目需要使用 Carbon 可以执行如下命令安装。

```
composer require nesbot/carbon
```

使用

直接通过 use 引用即可

```
use Carbon\Carbon;
```

获取当前时间

可以用 now() 方法获取当前的时间和日期。如果你想指定其他时区，也可以给 now() 方法传入一个参数指定。

```
echo Carbon::now(); // 2018-10-10 15:57:47
echo Carbon::now('America/Los_Angeles'); // 2018-10-10 01:00:30
```

除了 now() 方法外，Carbon 还提供了 today(), tomorrow(), yesterday() 等静态方法，但是它们的时间都是 00:00:00：

```
echo Carbon::today(); // 2018-10-10 00:00:00
echo Carbon::tomorrow(); // 2018-10-11 00:00:00
echo Carbon::yesterday(); // 2018-10-09 00:00:00
```

日期类型转换为字符串

默认情况下，Carbon 的方法返回的是一个日期时间对象。如果需要的是字符串，可以使用 toDateString() 或 toDateTimeString() 方法转换：

```
echo Carbon::now()->toDateString(); // 2018-10-10
echo Carbon::now()->toDateTimeString(); // 2018-10-10 16:13:28
```

nl2br() 函数

在字符串中的新行 (\n) 之前插入换行符：

```
echo nl2br("One line.\nAnother line.");
```

以上代码的浏览器输出：

```
One line.  
Another line.
```

五个 Laravel Blade 指令

1. 检测用户是否认证

你可以通过验证用户是否为空来检测其是否认证：

```
if(auth()->user())  
    // 用户已认证  
endif
```

然而，Laravel 自带的 Blade 命令可以更简洁地实现相同的功能：

```
auth  
    // 用户已认证  
endauth
```

2. 检测用户是否为访客

与认证相反，我们可以用 auth 辅助函数的 guest() 方法来检测用户是否为访客：

```
if(auth()->guest())  
    // 用户未认证  
endif
```

不过 Laravel 也为此提供了 @guest 命令：

```
guest  
    // 用户未认证  
@endguest
```

我们也可以使用 else 语句来组合这两个命令：

```
@guest  
    // 用户未认证  
@else  
    // 用户已认证  
@endguest
```

3. 如果第一个视图存在则引入，否则引入第二个

构建多主题站点可能会有一个文件如果存在就引入，否则就引入另一个的需要，你可以简单地使用条件判断来实现：

```
@if(view()->exists('first-view-name'))
    @include('first-view-name')
@else
    @include('second-view-name')
@endif
```

不过还是有一个更简洁直观的命令来做这件事：

```
@includeFirst(['first-view-name', 'second-view-name']);
```

4. 根据条件引入视图

当你只想在一定逻辑的基础上（如：一个已通过认证的用户）添加一些内容的时候，根据条件引入视图就非常有用。

你可以使用 @if 条件来这样写：

```
@if($post->hasComments())
    @include('posts.comments')
@endif
```

我们可以只用一行命令 @includeWhen 来做到：

```
@includeWhen($post->hasComments(), 'posts.comments');
```

5. 引入一个存在的视图

如果你有自定义主题系统或者你需要动态地创建 Blade 视图，那么检查文件是否存在就是必须要做的。

可以在辅助函数 view() 上调用 exists 方法：

```
@if(view()->exists('view-name'))
    @include('view-name')
@endif
```

也可以使用 Blade 命令 includelf 来处理：

```
@includeIf('view-name')
```

你可以通过 Blade 官方文档 了解更多实用的技巧来优化你 Laravel 项目里的前端模板。
重构快乐！

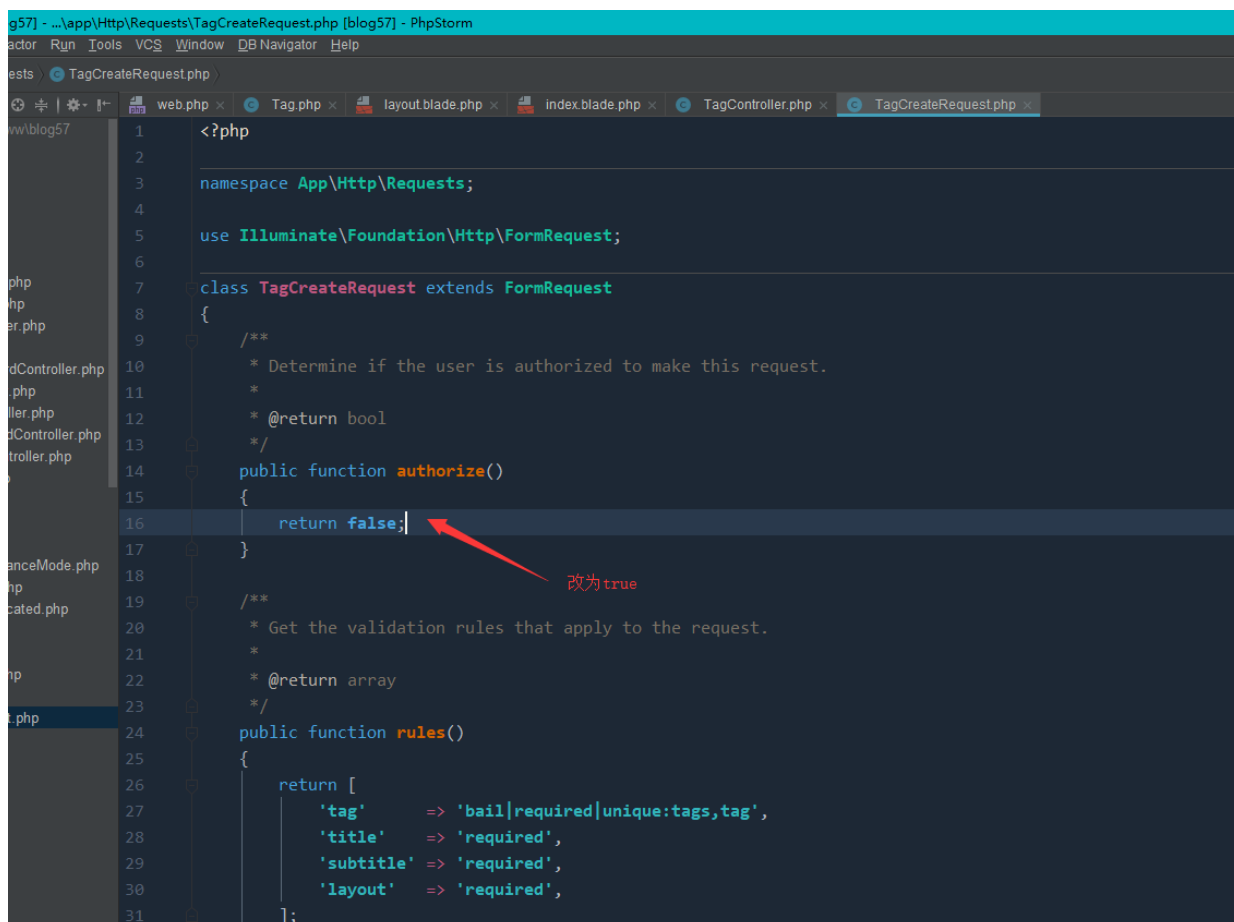
提交表单 报错

This action is unauthorized.?

403

This action is unauthorized.

GO HOME



The image shows a web browser displaying a 403 error message: "This action is unauthorized." with a "GO HOME" button. Below the browser, a code editor (PhpStorm) is open, showing the file `TagCreateRequest.php`. The code defines a `TagCreateRequest` class extending `FormRequest`. The `authorize()` method is highlighted with a red arrow pointing to the `return false;` line, with a red text annotation "改为true" (change to true) next to it. The `rules()` method is also visible, defining validation rules for 'tag', 'title', 'subtitle', and 'layout'.

```
<?php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class TagCreateRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return bool
     */
    public function authorize()
    {
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array
     */
    public function rules()
    {
        return [
            'tag' => 'bail|required|unique:tags,tag',
            'title' => 'required',
            'subtitle' => 'required',
            'layout' => 'required',
        ];
    }
}
```

在Laravel中，使用migration作为数据库的版本控制工具，当需要对已存在的数据表作更改，需要额外引入doctrine/dbal扩展。

Doctrine

```
composer require doctrine/dbal
```

MarkDown包管理工具SmartyPants

```
composer require michelf/php-markdown
composer require michelf/php-smartypants
```

引入 Selectize.js 和 Pickadate.js

下面为后台文章功能引入两个前端 JS 资源，我们使用 NPM 下载资源，然后通过 Laravel Mix 引入。

使用 NPM 下载资源—使用root或者administrator运行

首先是 Selectize.js。Selectize.js 是一个基于 jQuery 的 UI 控件，对于标签选择和下拉列表功能非常有用。我们将使用它来处理文章标签输入。使用 NPM 下载 Seletize.js：

```
npm install selectize --save-dev
```

接下来下载 Pickadate.js。Pickadate.js 是一个轻量级的 jQuery 日期时间选择插件，日期时间插件很多，选择使用 Pickadate.js 的原因是它在小型设备上也有很好的体验。下面我们使用 NPM 下载安装 Pickadate.js：

```
npm install pickadate --save-dev
```

使用 Laravel Mix 管理前端资源

现在相应的前端资源已经下载好了，接下来我们使用 Laravel Mix 来管理这些资源。编辑根目录下的 webpack.mix.js 配置文件如下：

```
const mix = require('laravel-mix');

/*
|-----
|
| Mix Asset Management
|-----
|
| Mix provides a clean, fluent API for defining some Webpack build steps
| for your Laravel application. By default, we are compiling the Sass
| file for the application as well as bundling up all the JS files.
|
*/

mix.js('resources/js/app.js', 'public/js')
```

```
.sass('resources/sass/app.scss', 'public/css');

**Selectize 暂不支持 Bootstrap 4**
mix.combine([
    'node_modules/selectize/dist/css/selectize.css',
    'node_modules/selectize/dist/css/selectize.bootstrap3.css'
], 'public/css/selectize.default.css');

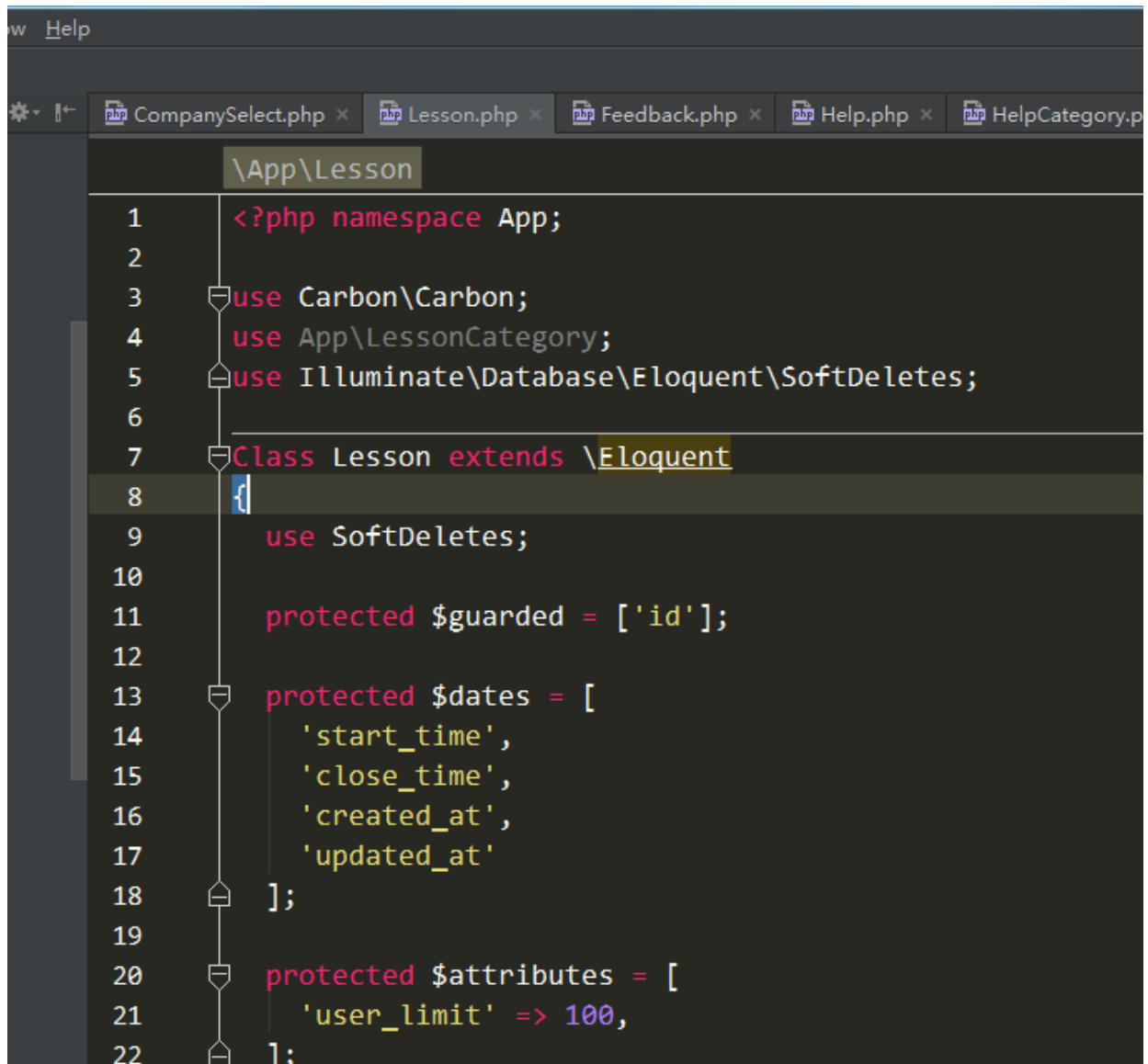
mix.combine([
    'node_modules/pickadate/lib/compressed/themes/default.css',
    'node_modules/pickadate/lib/compressed/themes/default.date.css',
    'node_modules/pickadate/lib/compressed/themes/default.time.css',
], 'public/css/pickadate.min.css');

mix.copy('node_modules/selectize/dist/js/standalone/selectize.min.js',
    'public/js/selectize.min.js');

mix.combine([
    'node_modules/pickadate/lib/compressed/picker.js',
    'node_modules/pickadate/lib/compressed/picker.date.js',
    'node_modules/pickadate/lib/compressed/picker.time.js'
], 'public/js/pickadate.min.js');
```

运行 `npm run dev` 重新编译前端资源。

laravel-模型中各种属性详解



```
1 <?php namespace App;
2
3 use Carbon\Carbon;
4 use App\LessonCategory;
5 use Illuminate\Database\Eloquent\SoftDeletes;
6
7 class Lesson extends \Eloquent
8 {
9     use SoftDeletes;
10
11     protected $guarded = ['id'];
12
13     protected $dates = [
14         'start_time',
15         'close_time',
16         'created_at',
17         'updated_at'
18     ];
19
20     protected $attributes = [
21         'user_limit' => 100,
22     ];
```

1. \$guarded属性, \$fillable属性

\$guarded属性一般是和\$fillable对应的，不是一起存在但是互相使用，他们都是laravel的批量赋值方法create()的，一个设置属性参数，有点这个意思。

在create方法收集数据赋值的时候

```
$flight = App\Flight::create(['name' => 'Flight 10']);
```

\$fillable就像是可以被赋值属性的“白名单”，还可以选择使用\$guarded。\$guarded属性包含你不想被赋值的属性数组。所以不被包含在其中的属性都是可以被赋值的，因此，\$guarded方法就像“黑名单”。当然，你只能同时使用其中一个——而不是一起使用：

\$fillable属性里面的字段被填上，说明这个字段是可以赋值的，其他的所有属性不能被赋值

\$guarded属性里面的字段被填上，说明这个字段不可以赋值，其他的所有属性都能被赋值

所有\$guarded相对来说在模型中出现频率比那个高。

2.\$dates属性

```
protected $dates = [
    'start_time',
    'close_time',
    'created_at',
    'updated_at'
];
```

里面所包含的字段，就是当使用这个属性的时候，可以直接后面跟着carbon类时间操作的任何方法，例如一个模型：

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class User extends Model{
    /**
     * 应该被调整为日期的属性
     *
     * @var array
     */
    protected $dates = ['created_at', 'updated_at', 'disabled_at'];
}
```

使用到这个属性disabled，那么这个属性在\$dates里面是存在的，所以他后面可以直接跟getTimestamp()方法，来各种处理。getTimestamp()方法是取时间戳的，他是carbon类下的兄弟。

```
$user = App\User::find(1);
return $user->disabled_at->getTimestamp();
```

如果你在\$dates里面将 disabled_at属性去除，OK，你在用getTimestamp()方法就不行了，失去了操作carbon类方法的能力

3.\$attributes属性

默认给数据库里的一个字段赋值

```
protected $attributes = [
    'user_limit' => 100,
];
```

默认给这个模型表的user_limit字段附上100的值

4.\$timestamps属性

laravel默认会在create()方法创建添加数据的时候，将create_at字段更新，如果是进行修改操作，

将会更新updated_at属性里面的值。

如果将

```
public $timestamps = false;
```

设为假的话，表示create方法执行时，不会对create_at和updated_at修改

使用 Clean Blog

Clean Blog 是 Start Bootstrap 提供的一个免费博客模板，本节我们将使用该模板美化博客前台页面。

使用 NPM 获取 Clean Blog

```
npm install startbootstrap-clean-blog --save-dev
```

使用 Laravel Mix 管理 Clean Blog

在 resources/sass/app.scss 中引入 Clean Blog 的 Sass 文件：

```
// Clean Blog
```

```
@import "~startbootstrap-clean-blog/scss/clean-blog";
```

然后运行 `npm run dev` 重新编译前端资源，新添加的 Clean Blog 的 Sass 资源文件就会通过 Laravel Mix 编译合并到 public/css/app.css 中。

发送邮件

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.163.com
MAIL_PORT=465
MAIL_USERNAME=17611241050@163.com
MAIL_PASSWORD=bestmoon007 #授权码
MAIL_ENCRYPTION=ssl
MAIL_FROM_ADDRESS=17611241050@163.com
MAIL_FROM_NAME=MOON
```

```
//获取全部数据
$allData = $request->all();
//只获取name和age
$onlyData = $request->only('name','age');
//获取除了name的所有数据
$exceptData = $request->except('name');
```

laravel使用redis报错

```
Predis \ Connection \ ConnectionException
```

```
`AUTH` failed: ERR Client sent AUTH, but no password is set [tcp://127.0.0.1:6379]
```

database需要改成下面

```
'redis' => [
    'client' => 'predis',

    'default' => [
        'host'      => env('REDIS_HOST', '127.0.0.1'),
//        'password' => env('REDIS_PASSWORD', null),
        'port'      => env('REDIS_PORT', 6379),
        'parameters' => [
            'password' => env('REDIS_PASSWORD', null)
        ],
        'database'  => env('REDIS_DB', 0),
    ],

    'cache' => [
        'host'      => env('REDIS_HOST', '127.0.0.1'),
        'password' => env('REDIS_PASSWORD', null),
        'port'      => env('REDIS_PORT', 6379),
        'database' => env('REDIS_CACHE_DB', 1),
    ],
],
```