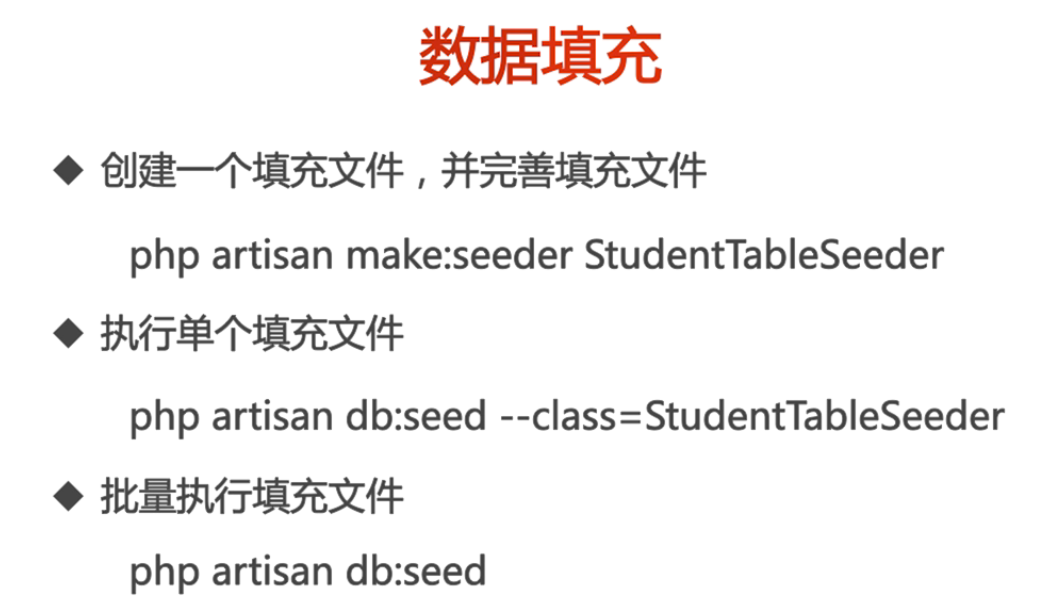
1.生成auth认证命令行:php artisan make:auth

2.数据迁移生成数据表:php artisan migrate

3.生成迁移文件:(1)php artisan make:migration create\_students\_table --create=students

4.生成迁移文件同时生成model:php artisan make:model Article -m

5.

1. 在使用post或者put等方法请求时，有时会报TokenMismatchException in VerifyCsrfToken.php line 67错误。原因是laravel默认开启了防CSRF。一种是在请求时将token值也提交过去，另一种是在防CSRF时排除所请求的路由。

eg:<form action="photo/12" method="post">

<?php echo method\_field('PUT'); ?>

<?php echo csrf\_field(); ?>

或者

{{ csrf\_field() }}

<input type="submit" name="提交" />

</form>

使用AJAX请求时：

<meta name="csrf-token" content="{{ csrf\_token() }}">

$.ajaxSetup({

headers: {

'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')

}

});

1. 文件上传

public function uploads(Request $request)

{

if($request->isMethod('POST')){

$file = $request->file('source');

//文件是否上传成功

if($file->isValid()){

//原文件名

$originaName=$file->getClientOriginalName();

//扩展名

$ext=$file->getClientOriginalExtension();

//MimeType

$type=$file->getClientMimeType();

//临时绝对路径

$realPath=$file->getRealPath();

$fileName=date('Y-m-d-H-i-s') . '-' . uniqid() . '.' . $ext;

$bool=Storage::disk('uploads')->put($fileName,file\_get\_contents($realPath));

var\_dump($bool);

exit;

}

}

return view('student/uploads');

}

1. 邮件发送
2. /config/mail.php 配置全局变量

'from' => [

'address' => env('MAIL\_FROM\_ADDRESS', '462369233@qq.com'),

'name' => env('MAIL\_FROM\_NAME', 'ksx'),

],

1. /.env 文件配置

MAIL\_DRIVER=smtp

MAIL\_HOST=smtp.qq.com

MAIL\_PORT=465

MAIL\_USERNAME=462369233@qq.com

MAIL\_PASSWORD=\*\*\*\*\*\*\*\*\*\*\*\*\*（qq邮箱为授权码，其他邮箱为密码）

MAIL\_ENCRYPTION=ssl

1. //发送文本邮件

Mail::raw('邮件测试',function ($message){

$message->from('462369233@qq.com','ksx');

$message->subject('测试内容');

$message->to('251882635@qq.com');

});

//发送html格式邮件

Mail::send('student.mail',['name'=>'ksx','phone'=>'18819201898'],function ($message){

$message->to('903363777@qq.com');

});

1. 异常返回页面

public function error(){

$student=null;

if($student==null){

abort(‘503’);

}

}

异常返回页面，在对应的视图目录新建异常返回页面，eg:503.blade.php

1. 响应

//返回上一级页面

return redirect()->back();

//重定向并携带数据，通过session获取

return redirect('session2')->with('message','数据');

//响应json

return response()->json($data);

1. 中间件
2. Artisan 命令创建中间件：php artisan make:middleware \*\*\*\*
3. 默认方法体

public function handle($request, Closure $next)

{

return $next($request);

}

1. 分配中间件到路由

在 app/Http/Kernel.php 文件中分配给该中间件一个key

1. 分页
2. 控制器查找分页信息：$students=Student::paginate(2);
3. 模板页调用render方法：{{ $students->render() }}
4. url判断

Request::getPathInfo() == ‘/\*\*\*’

1. 控制器验证表单字段

//控制器验证

$this->validate($request,[

'Student.name'=>'required|min:2|max:20',

'Student.age'=>'required|integer',

'Student.sex'=>'required|integer'

],[

'required'=>':attribute 为必填项',

'integer'=>':attribute 为整数',

'min'=>':attribute 长度不符合要求',

'max'=>':attribute 长度不符合要求'

],[

'Student.name'=>'姓名',

'Student.age'=>'年龄',

'Student.sex'=>'性别'

]);

验证不通过抛出异常，并赋值给全集变量$errors.

1. validator验证

//2.Validator类验证 全局

$validator=\Validator::make($request->input(),[

'Student.name'=>'required|min:2|max:20',

'Student.age'=>'required|integer',

'Student.sex'=>'required|integer'

],[

'required'=>':attribute 为必填项',

'integer'=>':attribute 为整数',

'min'=>':attribute 长度不符合要求',

'max'=>':attribute 长度不符合要求'

],[

'Student.name'=>'姓名',

'Student.age'=>'年龄',

'Student.sex'=>'性别'

]);

if($validator->fails()){

//withErrors($validator)：带上错误信息

//withInput()：带上表单信息 模板中调用”{{ old('Student')['age'] }}”

return redirect()->back()->withErrors($validator)->withInput();

}

1. 时间格式

<http://carbon.nesbot.com/docs/>

1. 数据填充

<https://github.com/fzaninotto/Faker/>

在\database\factories\ModelFactory.php

$factory->define(App\Post::class,function(Faker\Generator $faker){

return [

'title'=>$faker->sentence(6),

'content'=>$faker->paragraph(10),

'user\_id'=>0,

];

});

使用php artisan tinker 创建数据

Factory(App\Post::class,20)->make(); 在界面显示，不写入数据库

Factory(App\Post::class,20)->create(); 20:创建数据的条数

1. 控制字符串的长度

/\*\*

@param1:内容

@param2:长度

@param3:’结尾’

\*/

str\_limit($post->content,100,’...’)

1. 数据注入

使用ORM对象直接注入数据：例

$post=Post::create(request(['content','title','user\_id']));

dd($post);

直接注入会报MassAssignmentException错误，需要在模型（Model）中设置

protected $guarded; //不可以注入的字段

protected $fillable=['user\_id','title','content']; // 可以注入的字段

1. 验证

//验证

$this->validate(\request(),[

'title'=>'required|string|max:100|min:5',

'content'=>'required|string|min:10',

]);

@if(count($errors)>0)

<div class="alert alert-danger" role="alert">

@foreach($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</div>

@endif

1. laravel 变量不转义（适用于html代码输出）

例：{!! $post->content !!}

1. Artisan 命令创建符号链接

php artisan storage:link

1. 权限policy

创建命令:php artisan make:policy xxxPolicy

注册策略类：\app\Providers\AuthServiceProvider.php中注册

1. 模型关联



1. elasticsearch搜索引擎

下载地址：<https://github.com/medcl/elasticsearch-rtf>

前提：java jdk>1.8+

1. 解压安装包
2. cmd进入D:\elasticsearch-rtf-master\elasticsearch-rtf-master\bin
3. elasticsearch-plugin list 查看插件
4. 如果需要删除插件，就删除plugin\目录下的插件，重新cmd键入elasticsearch-plugin list 查看插件
5. 启动 elasticsearch -d
6. 
7. 使用 composer 包管理器来安装 Scout：

composer require laravel/scout

在config\app.php添加：

Laravel\Scout\ScoutServiceProvider::class,

用artisan命令执行：

php artisan vendor:publish --provider="Laravel\Scout\ScoutServiceProvider"

8.使用 composer安装scout的es驱动：

composer require tamayo/laravel-scout-elastic

修改config\scout.php

'driver' => env('SCOUT\_DRIVER', 'elasticsearch'),

添加驱动：

'elasticsearch' => [

'index' => env('ELASTICSEARCH\_INDEX', 'laravel54'),

'hosts' => [

env('ELASTICSEARCH\_HOST', 'http://127.0.0.1:9200'),

],

],

1. 创建command命令：php artisan make:command ESInit

修改app\Console\Command\ESInit.php

//使用什么命令启动脚本

protected $signature = 'es:init';

//描述

protected $description = 'init laravel es for post';

在app\Console\Kernel.php中挂载

protected $commands = [

\App\Console\Commands\ESInit::class

];

使php artisan命令查看命令挂载成功



1. 安装guzzlehttp/guzzle 扩展

composer require guzzlehttp/guzzle

1. 修改app\Console\Command\ESInit.php

public function handle()

{

//创建template

$client=new Client();

$url=config('scout.elasticsearch.hosts')[0]. '/\_template/tmp';

//$client->delete($url);

$param = [

'json'=>[

'template' => config('scout.elasticsearch.index'),

'mappings' => [

'\_default\_' => [

'dynamic\_templates' => [

[

'strings' => [

'match\_mapping\_type' => 'string',

'mapping' => [

'type' => 'text',

'analyzer' => 'ik\_smart',

'fields' => [

'keyword' => [

'type' => 'keyword'

]

]

]

]

]

]

]

],

],

];

$client->put($url,$param);

//记录

$this->info("=======创建模板成功=======");

//创建index

$url = config('scout.elasticsearch.hosts')[0] . '/' . config('scout.elasticsearch.index');

//$client->delete($url);

$param=[

'json' => [

'settings' => [

'refresh\_interval' => '5s',

'number\_of\_shards' => 1,

'number\_of\_replicas' => 0,

],

'mappings' => [

'\_default\_' => [

'\_all' => [

'enabled' => false

]

]

]

]

];

$client->put($url,$param);

//记录

$this->info("=========创建索引成功=========");

}

1. 修改模型 -> post.php

use Searchable;

//定义索引里面的type

public function searchableAs()

{

return "post";

}

//定义有哪些字段需要搜索

public function toSearchableArray()

{

return [

'title'=>$this->title,

'content'=>$this->content,

];

}

1. 导入数据（开启php curl扩展）

artisan命令：php artisan scout:import "\App\Post"

1. 视图合成器
2. 将合成器放在AppServiceProvider.php

View::composer('layout.slidebar',function ($view){

$topics = \App\Topic::all();

//将数据注入

$view->with('topic',$topics);

});

1. 下载adminlte 后台管理模板

composer require "almasaeed2010/adminlte=~2.0"

<https://github.com/kesixin/AdminLTE>

1. 定义后台认证
2. 定义后台auth-provider，在\config\auth.php中’providers’中添加

'admins' => [

'driver' => 'eloquent',

'model' => App\AdminUser::class,

],

（2）定义后台auth-guards，在\config\auth.php中’guguards’中添加

'admin' => [

'driver' => 'session',

'provider' => 'admins',

],

1. Gate 用户授权
2. App\Providers\AuthServiceProvider 类中使用Gate定义

//注册Gate

$permissions = \App\AdminPermission::all();

foreach ($permissions as $permission) {

Gate::define($permission->name,function ($user) use($permission){

//use($permission) 接收外面传递的permission

return $user->hasPermission($permission);

});

}

1. 路由中middleware授权路由(sysyem:permission权限)

Route::group(['middleware' => 'can:system'],function (){

});

（3）blade模板中定义权限显示内容

@can("system")

......

@endcan

1. 队列



1. 修改配置文件.env 修改默认驱动

QUEUE\_DRIVER=database

1. 创建队列表

php artisan queue:table

会在\database\migrations\创建一个jobs文件

使用php artisan migrate创建jobs表

1. 创建任务

php artisan make:job SendMessage

会创建\jobs\SendMessage.php文件

use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

private $notice;

/\*\*

\* Create a new job instance.

\*

\* @return void

\*/

public function \_\_construct(\App\Notice $notice)

{

$this->notice = $notice;

}

/\*\*

\* Execute the job.

\*

\* @return void

\*/

public function handle()

{

//通知每个用户系统消息

$users = \App\User::all();

foreach ($users as $user)

{

$user->addNotice($this->notice);

}

}

1. 分发任务

在通知保存数据库成功之后分发任务

public function store()

{

$this->validate(request(), [

'title' => 'required|string',

'content' => 'required|string',

]);

$notice = \App\Notice::create(request(['title','content']));

dispatch(new \App\Jobs\SendMessage($notice));

return redirect('/admin/notices');

}

1. 启动队列

php artisan queue:work

1. 优化

（1）路由缓存

php artisan route:cache 不使用路由匿名函数

会生成\bootstrap\cache\routes.php文件

清空:php artisan route:clear

1. 配置缓存

php artisan config:cache

会生成\bootstrap\cache\config.php文件

清空：php artisan config:clear

1. 优化类加载

php artisan optimize

会在\vendor\composer\autoload\_classmap.php中添加类映射

清空：php artisan clear-compiled

1. 使用Laravel-debugbar 开发调试利器
2. 安装：如果laravel 版本小于5.5使用debugbar2.4版本

composer require barryvdh/laravel-debugbar:~2.4

（2）注册如下服务提供者：\config\app.php

Barryvdh\Debugbar\ServiceProvider::class,

1. 将debugbar配置文件提取

php artisan vendor:publish --provider="Barryvdh\Debugbar\ServiceProvider"

1. 预加载优化 使用（with/load）

public function index()

{

$posts = Post::orderby('created\_at', 'desc')->withCount(['zans','comments'])->with('user')->paginate(6);

//load 预加载

//$posts->load('user');

return view('post.index', compact('posts'));

}