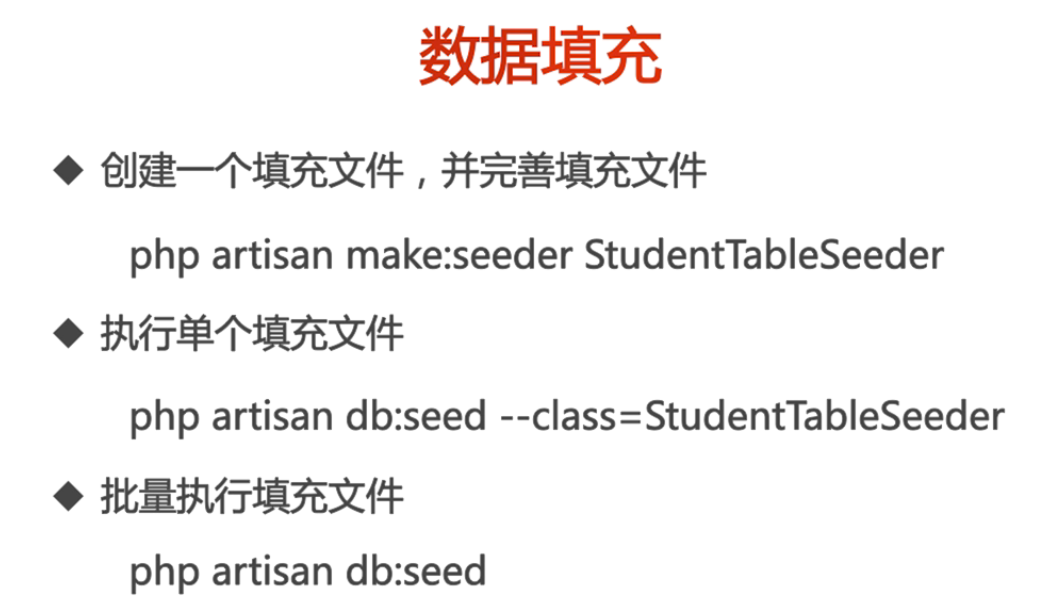
1.生成auth认证命令行:php artisan make:auth

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 !!}