Laravel

* 目录

/public/index.php // 入口

/config/\*.php // 全局配置文件

/routes/ // 路由配置目录

/app/http/ // 逻辑层

controller/ // 控制器目录

middleware/ // 中间件目录

/rersource/ // 资源文件目录

views/ // 视图目录

/vendor/ // 第三方扩展目录

laravel/freamework// 框架核心目录

.env // 全局快速配置文件

* artisan

php artisan down // 声明项目处于维护状态(503)

php artisan up // 开启项目

php artisan make:controller filename // 新建一个控制器

php artisan make:controller filename –resource // 新建一个资源控制器

php artisan make:middleware filename // 新建中间件

php artisan make:request filename // 新建一个表单请求过滤文件(类似中间件)

php artisan make:model filename // 新建一个模型类

* 路由配置(/routes/web.php)

Route::请求方式(‘路由地址/{参数}’, 回调函数|控制器地址);

->where(‘参数’, ‘正则表达过滤参数’)

->middleware(‘中间件’);

别名方式: Route::get(‘/user’,[‘as’=>’别名’, 回调函数]);

请求方式:get // get请求

post // post请求

any // 所有请求

match // 匹配部分请求：Route::match([‘请求方式’], ‘路由地址’, 回调函数|控制器);

Route::group([‘prefix’=>’统一前缀’, ‘middleware’=>’中间件名称’], function(){详细路由配置});

route(); // 通过别名获取指定路由的规则

资源文件路由

Route::resource(‘test, ‘资源控制器’);

index : GET /test

create: GET /test/create

store: POST /test

show: GET /test/1

edit: GET /test/1/edit

update: PUT/PATCH /test/1/

destroy: DELETE /test/1

* 函数助手

dd() = dump die;

Config::get(‘file.key’); // 获取config/\*.php的配置值

Config::set(‘key’, ‘value’) // 在config/\*.php 设置一个值

例如:Config::get(‘app.timezone’); // 获取config/app timezone的配置值

env(‘key’, ‘value’); // 获取.env的配置项，如果不存在则使用default

* 响应
  + 请求(Request $request)

$request ->path(); // 获取请求路径

$request ->url(); // 获取请求的路由

$request ->method(); // 获取请求方式

$request ->isMethod(‘post’); // 检测请求方式是否为post

$request ->all(); //获取所有参数

$request ->input(‘key’, ‘default’); // 获取指定的参数，如果key为空则使用default

$request ->only([‘key’, ’key’]); // 获取指定的参数

$request ->except([‘key’, ’key’]); // 获取指定参数之外所有的参数

$request ->has(‘key’); // 检测指定的参数是否存在

* + - 闪存(稍瞬即逝)

$request ->flash(); // 将请求参数存入闪存

$request ->flashOnly(); // 将指定的参数存入闪存

$request ->flashExcept(); // 排除指定的参数将剩下的存入闪存

return back() ->withInput(); // 取消页面的提交并返回闪存数据

return back(); // 阻止页面提交$request ->file(); // 获取上传的文件

// 文件上传

$request ->hasFile(‘file’); // 查看指定的文件是否被上传

$request ->file()

->extention() 或者

->getClientOriginalExtension(); // 获取文件后缀名

->isValid(); 验证上传错误

$request ->move(‘path’, ‘filename’); // 将上传的文件移动到path下起名为filename(没有后缀)

* + cookie和session(响应)

cookie

设置cookie

use Cookie; // 参考\config\app.php aliases[] 别名

设置：Cookie::queue(‘key’, ‘value’, lifetime);

获取：Cookie::get(‘key’);

响应cookie

$request ->cookie(‘key’);

return response() ->withCookie(‘key’, ‘value’, lifetime);

return response() ->json({}); // 响应返回json

session

session([‘key’, ‘value’]); // 设置一个session

session(‘key’); // 获取session

session() ->all(); // 获取所有session

session() ->has(‘key’); // 查看指定的session是否存在

session() ->push(‘key’, ‘value’); // 向同一个session里压如一个值

session() ->pull(‘key’); // 获取指定的session并删除

->with(‘key’,’value’);响应时设置session信息

* 数据库 use DB;
  + 增

DB::insert(“添加sql语句”);

DB::table(‘table’) ->insert([‘字段’=>’值’, ’字段’=>’值’]);

DB::table(‘table’) ->insert([[‘字段’=>’值’], [‘字段’=>’值’]]); // 直接插入，返回插入是否成功，可使用一维、二维数组形式的格式

DB::table(‘table’) ->insertGetId([‘字段’=>’值’]); // 插入数据并获取插入的ID (只能使用一维数组)

* + 删

DB::delete(‘删除sql语句’);

DB::table(‘table’) ->delete(); // 删除已选中的

* + 改

DB::update(‘修改sql语句’);

DB::table(‘table’) ->update([‘字段’=>’值’]); // 修改已选中的字段为值

* + 查

DB::select(‘查询sql语句’);

DB::statement(‘sql语句入删库等操作’);

DB::table(‘table’) ->get(); // 获取数据库所有数据，以对象形式返回

DB::table(‘table’)

->where(‘条件’)

->orWhere(‘逻辑或条件’) // 若条件为=则可以省略

->whereBetween(‘id’,[1,3]) // id在 1~3之间的

->orWhereBetwwen(‘id’, [4,9]) // 或者id在4~9之间的数据

->whereIn(‘id’,[1,4,7]) // id等于1,4,7的值

->limit() // 筛选指定的条数

->orderBy(‘id’,’desc’) // 通过id降序排序

->count() // 获取查询的数据数量

->max() // 获取该字段的最大值

->min() // 获取该字段的最小值

->avg() // 获取该字段的平均值

->first() // 获取单条数据

->pluck() // 获取指定列的数据

->join(‘table2’,’table.id’,’table2.id’) // 链表查询

->select() // 选择指定的字段

->get(); // 获取选中数据

* 视图(view)

引入 @include(); // 无法重写占位

继承 @extends(); // 继承指定的模板

占位符

@yield(“占位符名称”, ‘占位符值可以被重写’);

@section(“占位符名称”); @show 结束

@section(“重写占位符”); @endsection 结束重写

流程

@if @elseif @endif

输出

{{$demo}} // 输出demo（内容会被转换为实体）

{!!$demo!!} // 输出demo

{{time()}} // 输出时间戳

* 分页

DB：：table(‘test’) ->paginate(数量);

$data ->render() // 获取分页按钮

$data ->

* 加密(Hash加密)

Hash::make(‘password’) // 对password进行加密

* 表单实例
* 模型(php artisan make:model filename)

protected $table = 指定数据表

protected $primarykey = 主键

public $timestamp = 自动维护创建时间和修改时间字段

public function getStatusAttribute($value){ // 对取值进行处理

$sex = [1 =>’男’, 0 =>’女’]

return $sex[$value];

}

protected $fillable = [] = 查询字段

* 扩展
  + 自定义方法 (/app/Libaray/\*.php)

composer.json ->autoload ->files

自定义类 (/app/Libaray/\*.php)

composer.josn ->autoload ->classmap

composer dump-autoload (添加)

* debugbar(调试工具)

app.config ->引入命名空间

composer require barr

* composer require 外部文件
  + 参照packagelist.org官网包

Redis

use Cache || use Redis