php从5升级到7: curl -s http://php-osx.liip.ch/install.sh | bash -s 7.1

export PATH=/usr/local/php5/bin:$PATH

下载mysql： http://www.mysql.com/downloads/mysql

Laravel框架：

安装composer：curl -sS https://getcomposer.org/installer | php

sudo mv composer.phar /usr/local/bin/composer

sudo chmod a+x /usr/local/bin/composer

安装laravel：composer global require "laravel/installer"

laravel new如果报错解决：nano ~/.bashrc 插入alias laravel='~/.composer/vendor/bin/laravel' 终端source ~/.bashrc

再laravel new projectName

创建指定版本的laravel项目composer create-project --prefer-dist laravel/laravel blog "5.1.\*"

本地可以正常打开却在远程服务器上不能打开：可能是权限问题：sudo chown -R :www /var/vhosts/blog.hnbwqr.xyz其中www是[www.conf中的用户](http://www.conf中的用户)；sudo chmod -R 775 /var/vhosts/blog.hnbwqr.xyz/storage sudo chmod -R 775 /var/vhosts/blog.hnbwqr.xyz/bootstrap/cache 修改.env

正确上传git做法：不要git add –f –A，就是git add –A ； 服务器端copy .env.example并修改；composer update ；如果出现The only supported ciphers are AES-128-CBC and AES-256-CBC错误，就php artisan key:generate php artisan config:clear php artisan config:cache

如果出现view [home] not found 再php artisan config:clear

使用laravel：

本地启动：php artisan serve

获取环境变量：App::environment();

判断环境变量是否是某个值：App::environment('local', 'staging')

访问配置值：config('app.timezone');

设置配置值：config(['app.timezone' => 'America/Chicago']);

显示维护模式php artisan down 关闭维护模式php artisan up

路由：/app/Http/routes.php中Route::get('/', function () { return view('welcome');});则浏览器中localhost/public即返回/resources/views/welcome.blade.php中的内容

Route::get('now', function () {return “…”});相应的路由即localhost/public/now

Post请求实例(要token)：Route::get('/testPost',function(){$csrf\_token = csrf\_token();$form = <<<FORM<form action="/hello" method="POST"><input type="hidden" name="\_token" value="{$csrf\_token}"><input type="submit" value="Test"/></form>FORM;return $form;}); Route::post('/hello',function(){return "Hello Laravel[POST]!";});

match方法匹配多种请求方式：Route::match(['get','post'],'/hello',function(){return "Hello Laravel!";});

any方法匹配任意请求方式Route::any('/hello',function(){return "Hello Laravel!";});

路由参数Route::get('/hello/{name}',function($name){return "Hello {$name}!";});

注意：这样如果return view可能会造成css失效，具体原因还不清楚

路由参数约束 Route::get('/hello/{name?}',function($name="Laravel"){return "Hello {$name}!";})->where('name','[A-Za-z]+'); 没有参数用默认值

设置全局约束：/app/RouteServiceProvider.php的boot方法中public function boot(Router $router){$router->pattern('name','[A-Za-z]+');parent::boot($router);}

路由重定向：Route::get('/hello/laravelacademy/{id}',['as'=>'academy',function($id){return 'Hello LaravelAcademy '.$id.'！';}]); Route::get('/testNamedRoute/{id}',function($id){

return redirect()->route('academy',['id'=>"{$id}"]);});

路由分组：

中间件：Route::group(['middleware'=>'test'],function(){Route::get('/write/laravelacademy', function(){ });}); 创建中间件：php artisan make:middleware TestMiddleware 在app/Http/TestMiddleware中：public function handle($request, Closure $next){if ($request->input('age')<18)return redirect()->route('refuse'); return $next($request); } 在Kernel.php中routeMiddleware加上'test' => \App\Http\Middleware\TestMiddleware::class,

命名空间：Route::group(['namespace' => 'LaravelAcademy'], function(){// 控制器在 "App\Http\Controllers\LaravelAcademy" 命名空间下 Route::group(['namespace' => 'DOCS'], function() { // 控制器在 "App\Http\Controllers\LaravelAcademy\DOCS" 命名空间下 }); });

子域名：Route::group(['domain'=>"{service}.localhost"],function(){Route::get ('/write',function($service){return "...{$service}...";});});

路由前缀：Route::group(['prefix'=>'laravelacademy/{version}'],function(){Route::get('write', function($version){return "...{$version}...";});});

CSRF攻击

Route::get('testCsrf',function(){$csrf\_field = csrf\_field();$html = <<<GET<form method= "POST" action="/testCsrf">{$csrf\_field}<input type="submit" value="Test"/></form>GET; return $html;}); Route::post('testCsrf',function(){return 'Success!';});如果没有csrf\_field则出现异常

排除制定URL的CSRF验证：app/Http/Middleware/VerifyCsrfToken.php中protected $except = ['testCsrf'];

中间件:

中间件参数：Route::group(['middleware'=>'test:male'],function(){Route::get ('/write/laravelacademy',function(){ //使用Test中间件 }); });Route::get('/age/refuse', ['as'=>'refuse',function(){ return "18岁以上男子才能访问！"; }]); 在中间件中public function handle($request, Closure $next, $gender){ if($request->input('age')>=18 && $gender==$request->input('gender')){return $next($request);}return redirect()->route ('refuse');} 其中test::male中的male就是默认的gender

控制器：php artisan make:controller PostController在app/Http/Controllers下生成

Route::resource('post','PostController'); <http://localhost:8000/post则进入index方法>

post访问进入store方法

Cache::increment('post\_id',1);自增1

Cache::decrement('post\_id',1); 删除

Route::controller在5.4中用Route::resource，但是只能用基本的create、show等路由，除非在web.php中用@设置

获取path：$uri = $request->path(); 获取uri：$uri = $request->path(); 获取method：$request->method() 判断参数是否存在$request->has('hello')

将请求数据暂时保存到session中: $request->flash() 从session中获取并清空$request->old();

获取cookie：$request->cookie('website'); 添加cookie：$response->withCookie(cookie ('website','LaravelAcademy.org',1));

上传文件相关：$request->hasFile('file') $file = $request->file('file'); $file->isValid() 返回public目录下的images路径(没有返回””)：$destPath = public\_path('images'); file\_exists($destPath) mkdir($destPath,0755,true); $file->move($destPath,$filename)

//page

响应：

use Illuminate\Http\Response; Route::get('testResponse',function(){$content = 'Hello LaravelAcademy！';$status = 500;$value = 'text/html;charset=utf-8';return response($content,$status)->header ('Content-Type',$value);});

加cookie：在header后->withCookie('site','LaravelAcademy.org')

cookie有效期和作用域：->withCookie('site','LaravelAcademy.org',30,'/','laravel.app');

不加密该cookie：在app/Http/Middleware/EncryptCookies.php中将cookie名加入：protected $except = ['site'];

视图响应：Route::get('testResponseView',function(){$value = 'text/html;charset=utf-8';return response()->view('hello',['message'=>'Hello LaravelAcademy'])->header('Content-Type',$value);});在resources/views/hello.blade.php中echo "{$message}";

返回json数据格式：Route::get('testResponseJson',function(){return response()->json(['name'=>'LaravelAcademy','passwd'=>'LaravelAcademy.org']);});

文件下载Route::get('testResponseDownload',function(){return response()->download (realpath(base\_path('public')).'/model.html','model.html');});

重定向Route::get('dashboard', function () {return redirect('testResponseDownload');});

重定向到上一个位置return back()->withInput();

重定向到控制器方法Route::resource('post','PostController');Route::get ('testResponseRedirect',function(){return redirect()->action('PostController@show',[1]);});

带一次性session数据的重定向：Route::post('user/profile', function () { return redirect ('dashboard')->with('status', 'Profile updated!');});

视图间共享数据：在app/Providers/AppServiceProvider的boot方法中加入view()->share('sitename','Laravel学院');然后可以在/resources/views中php文件中用"{$sitename}"访问

绑定一些数据到视图中view()->composer(['hello','home'],function($view){$view->with ('user',array('name'=>'test','avatar'=>'/path/to/test.jpg'));}); views中通过{{$user['name']}}访问 或所有视图：用'\*'替换['hello','home']

服务容器：

$this->app->singleton('HelpSpot\API', function ($app) {return new HelpSpot\API($app-> make('HttpClient'));});其中HelpSpot\API为接口，HttpClient为实例

绑定一个已存在的对象实例到容器$api = new HelpSpot\API(new HttpClient); $this->app-> instance('HelpSpot\Api', $api);

同接口不同类：$this->app->when(PhotoController::class)

监听服务容器解析对象：$this->app->resolving(function ($object, $app) {…}); 或 $this->app->resolving(HelpSpot\API::class, function ($api, $app) {…});

创建并使用服务(相当于接口实例方法?)：

1.创建接口interface TestContract{public function callMe($controller);}

2.创建服务实例class TestService implements TestContract{public function callMe($controller) {dd('Call Me From TestServiceProvider In '.$controller);}}

3.创建服务提供者provider：public function register(){$this->app->singleton('test',function() {return new TestService();}); $this->app->bind('App\Contracts\TestContract',function() {return new TestService();};)用到TestContract就会返回一个service实例

4.注册服务提供者：在config/app.config中'providers' => [App\Providers\ TestServiceProvider::class,],

5.调用服务：public function \_\_construct(TestContract $test){$this->test = $test;} public function index(){$this->test->callMe('TestController');}

门面(相当于接口类方法?)：

1.在app/Facades/Test.php中：class Test{public function doSomething(){echo ' doSomething';}}

2.在app/Facades/TestClass.php中,门面类绑定到Test：class TestClass extends Facade{protected static function getFacadeAccessor(){return 'test';}}

3.在provider中绑定Test类到服务容器public function register(){$this->app->singleton ('test',function(){return new Test;});}

4.在config/app.php中注册门面类名'aliases' => ['TestClass' => App\Facades\TestClass::class,],

5.调用TestClass::doSomething();

配置数据库：config/database.php和.en都要改,再重启服务器

修改数据库密码# mysql -uroot –p Enter password: 【输入原来的密码】 mysql>use mysql; mysql> update user set password=passworD("test") where user='root'; mysql> flush privileges; mysql> exit;

插入数据：DB::insert('insert into users (id, name, email, password) values (?, ?, ? , ? )',

[null, 'Laravel','laravel@test.com','123']);

查询和查看：$user = DB::select('select \* from users where id = ?', [1]); dd($user);

更新: $affected = DB::update('update users set name="LaravelAcademy" where name = ?', ['Academy']); echo $affected;

删除DB::delete('delete from users');

删除表，DDL语句：DB::statement('drop table users');

终端同步表操作:php artisan migrate

事务; DB::transaction(function () {DB::table('users')->update(['id' => 1]);DB::table('posts')-> delete(); });

监听:在app/Providers/AppServiceProvider.php的boot方法中:

DB::listen(function($sql) { dump($query->bindings);});

事务：DB::transaction(function () {DB::table('users')->update(['name' => '1tt']);DB::rollback ();DB::table('posts')->delete();}); 或 DB::beginTransaction();DB::table ('users')-> update(['name' => 'test']);DB::rollback();DB::table('posts')->delete();DB::commit();

使用查询构建器: echo DB::table('users')->insertGetId( ['name'=>'Laravel-Academy','email'=> 'laravelacademy@test.com','password'=>'456']);

更新: echo $affected = DB::table('users')->where('name','Laravel-Academy')->update (['password'=>'123']);

删除: echo DB::table('users')->where('id', '>', 3)->delete(); DB::table('users')->truncate();

查询: dd(DB::table('users')->get()); dd(DB::table('users')->select('name','email')->get());

dd(DB::table('users')->where('name','Laravel')->first());

组块: DB::table('users')->orderBy('name')->chunk(2,function($users){foreach($users as $user){echo $user->id.'<br>'; if ($user->id=='3') {return false;}} });两个记录接两个记录分块查询

内连接: $users = DB::table('users')->join('posts','users.id','=','posts.user\_id')->get();

左连接(左连接的结果集指定的左表的所有行，如果左表的某行在右表中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为空值):

$users = DB::table('users')->leftJoin('posts','users.id','=','posts.user\_id')->get();

闭包条件: $users = DB::table('users')->join('posts',function($join){$join->on('users.id','=', 'posts.user\_id')->where('posts.user\_id','>',1);})->get();

联合查询: $users = DB::table('users')->where('id','<',2); $users = DB::table('users')->where ('id','>',3)->union($users)->get();

其他条件查询orWhere orderBy groupBy having 分页skip(0)->take(3) count max

通过迁移重建数据库php artisan migrate:refresh

Schema::rename 更新表结构Schema::table <http://laravelacademy.org/post/6964.html>

->nullable()->change() renameColumn dropColumn

创建填充器php artisan make:seeder UserTableSeeder

填充：在DatabaseSeeder的run方法中写入insert，终端php artisan db:seed

模型工厂factory('App\User', 50)->create()->each(function($u) {$u->posts()->save(factory ('App\Post')->make());});

ORM模型:

php artisan make:model Models/Post 指定表名(在Post.php中) public $table = 'posts'; public $primaryKey = 'id'; public $timestamps = false; protected $dateFormat = 'U';

$posts = Post::where('id','<',3)->orderBy('id','desc')->take(1)->get(); echo $posts[0]->title;

用模型方式添加数据$post = new Post; $post->title = 'test 4'; $post->content = 'test content';

if($post->save()){ echo '添加文章成功！'; }else{ echo '添加文章失败！'; }

批量赋值：$post = Post::create(Input::all()); 过滤$guarded或$fillable save不受影响

firstOrCreate和firstOrNew：查找匹配记录，没有找到则创建

更新数据: $post = Post::find(0); $post->title = 'test 1 title'; $post->save();

update方法更新数据$post = Post::find(6); $post->update($input)

删除$post->delete() Post::destroy(5);其中5是id 或Post::destroy([1,2,3,4,5]);

用查询构造器: Post::where('user\_id',1)->delete();

软删除

php artisan make:migration alter\_posts\_deleted\_at --table=posts

Schema::table('posts', function (Blueprint $table) {$table->softDeletes();});

use Illuminate\Database\Eloquent\SoftDeletes; 在类里面use SoftDeletes; 注意不要教程中的public $dateFormat = 'U';

调用$post = Post::find(1); $post->delete(); if($post->trashed()){echo '软删除成功！';dd($post);}else{echo '软删除失败！';}

查询结果包含软删除：$posts = Post::withTrashed()->get();

查看软删除：$posts = Post::onlyTrashed()->get();

软删除恢复Post::withTrashed()->find(1)->restore(); 恢复关联模型查询->history()->restore();

强制删除$post = Post::find(1); $post->forceDelete();

查询作用域，将一些查询封装到模型中

Post.php中public function scopePopular($query){return $query->where('views','>=',100);}

调用: $posts = Post::popular()->orderBy('views','desc')->get();

动态传参: 新增public function scopeStatus($query,$status=1){return $query->where ('status',$status);} 调用: $posts = Post::popular()->status(1) ->get();

模型事件监听: Post::saving(function($post){echo 'saving event is fired<br>';}); 其他: creating、created、updating、updated、saving、saved、deleting、deleted

如果在creating中return false，则不会执行created、deleted等操作

模型创建总结:

1.迁移文本php artisan make:migration create\_user\_accounts\_table --create=user\_accounts注意加s

2.迁移文本:up方法中:Schema::create('user\_accounts', function (Blueprint $table) {$table->increments('id'); $table->increments('user\_id');}); down中: Schema::drop('user\_accounts');其中user\_id不可少

3.迁移: php artisan migrate

4. 模型php artisan make:model Models/UserAccount不加s

关联,在User.php中:(一对一) public function account(){return $this->hasOne ('App\Models\UserAccount');} 调用$account = User::find(1)->account; dd($account);

也可以return $this->belongsTo('App\User','user\_id','id');其中后两个参数不加则用默认值,依次是外键、本键

一对多: return $this->hasMany('App\Models\Post');

多对多: belongsToMany('App\Models\Role'); 中间字段: echo $role->pivot->role\_id.'<br>';

远层一对多: hasManyThrough('App\Models\Post','App\User');第二个参数是中间对象类名,第三个参数是中间对象表示所在对象的建名，第四个参数是目标对象在中间对象的键名

多态关联: 在Post.php和Video.php中morphMany('App\Models\Comment','item'); 在comments表中item\_id，item\_type字段下有App\Models\Video和App\Models\Post； 其中item\_id对应videos表中的id, item\_type则对应Video

获取对应节点:在Comment.php中public function item(){return $this->morphTo();}调用: $comment = Comment::find(1); $item = $comment->item; dd($item);

多对多多态关联：在Video中public function tags(){return $this->morphToMany ('App\Models\Tag','taggable');}默认第三个参数是关联关系表名,第四个参数是关联关系表中源id，第五个参数是目标id

按标签查询节点：$tag = \App\Models\Tag::find(1); $posts = $tag->videos; dd($posts);

自定义包的开发

<http://laravelacademy.org/post/1219.html> 注：composer.json不是系统根目录下的那个，psr-4在里面已经存在，不要另外新加

session配置: config/session.php

创建session表：php artisan session:table php artisan migrate

用户认证: php artisan make:auth

生成注册和登录视图(resources/views/auth和resources/views/layouts(基础布局文件)), 认证路由和HomeController, 如果注册或登录成功后仍然返回login页面，需要创建session表

自定义登录用户采用的字段public function username(){return 'name';} 注：改完这个还要在view中将相应原来是email的地方改成name

guard的作用？ //page

RegisterController 的 validator 方法包含了新用户注册的验证规则

RegisterController的create方法创建新的user记录

显示登录特定次数后的错误: @if (count($errors) > 0)<div class="alert alert-danger"><ul style="color:red;">@foreach ($errors->all() as $error)<li>{{ $error }}</li>@endforeach</ul> </div>@endif

获取用户: use Illuminate\Support\Facades\Auth; $user = Auth::user();

返回认证用户实例use Illuminate\Http\Request; $request->user()

要判断用户是否登录到应用use Illuminate\Support\Facades\Auth; Auth::check()

路由认证Route::get('profile', 'ProfileController@index')->middleware('auth'); 或者在控制器中public function \_\_construct(){$this->middleware('auth');}

指定某个guard：public function \_\_construct(){$this->middleware('auth:api');

手动认证用户//?lymark

提供登录认证而不用设置专门的登录页面Route::get('profile', function() {…})->middleware('auth.basic');

重置密码报错：Expected response code 250 but got code 530 , … 改了.en里的东西需要重启aritisan 注意端口MAIL\_PORT=25，改完这个也要重启artisan

第三方登录 github

1. composer require laravel/socialite

2. config/app.php中，providers中Laravel\Socialite\SocialiteServiceProvider::class,

3. config/services.php中'github' => ['client\_id' => 'app client\_id','client\_secret' => ' app client\_secret','redirect' => 'http://laravel.app:8000/auth/github/callback']其中app client\_id是github创建的应用id：https://github.com/settings/developers

4. 在登录控制器中public function redirectToProvider(){return Socialite::driver('github') ->redirect();} public function handleProviderCallback(){$user = Socialite::driver('github')-> user(); if(!User::where('github\_id',$user->id)->first()){ $userModel = new User;$userModel-> github\_id = $user->id; $userModel->email = $user->email; … ;$userModel->save();} $userInstance = User::where('github\_id',$user->id)->firstOrFail();Auth::login ($userInstance);echo $user->name.'登录成功!';}

5. 路由Route::get('auth/github', 'Auth\LoginController@redirectToProvider'); Route::get('auth/github/callback', 'Auth\ LoginController @handleProviderCallback');

6. 访问http://localhost:8000/auth/github

//page 支付

认证api: //lymark

安装passport：

1. cd到项目根目录,

2. composer config -g repo.packagist composer <https://packagist.phpcomposer.com>

3. composer require laravel/passport

在config/app.php的providers数组中Laravel\Passport\PassportServiceProvider::class,

API认证（Passport）//?lymark: <http://laravelacademy.org/post/6813.html#toc_2>

用户授权gates和policies//?lymark: <http://laravelacademy.org/post/6823.html>

加密解密encrypt decrypt Crypt::encryptString decryptString

哈希: Hash::make($request->newPassword) 原生字符串和给定哈希是否相等:Hash::check('plain-text', $hashedPassword)

重置密码//?lymark <http://laravelacademy.org/post/6837.html> 发不出邮件

查看所有artisan命令：php artisan list

帮助：php artisan help migrate php artisan tinker//?lymark

编写命令: php artisan make:command SendEmails会在 app/Console下创建command目录

php artisan list 或 php artisan help 命令时显示: Artisan::command('build {project}', function ($project) {$this->info("Building {$project}!");})->describe('Build the project');

参数protected $signature = 'email:send {user}';

选项protected $signature = 'email:send {user} {--queue=}'; php artisan email:send 1 --queue=default

数组参数email:send {user\*} 用法php artisan email:send foo bar

定义一个期望输入数组的选项email:send {user} {--id=\*} 用法php artisan email:send --id=1 --id=2

加上描述protected $signature = 'email:send {user : The ID of the user} {--queue= : Whether the job should be queued}';

获取参数：在handle方法中$userId = $this->argument('user'); 注：要在handle中起作用需要在Kernel.php中protected $commands = ['App\Console\Commands\SendEmails'];

获取数组方式参数$arguments = $this->arguments();

获取选项值：$queueName = $this->option('queue'); $options = $this->options();

提示用户提供输入$name = $this->ask('What is your name?');

密码类型输入$password = $this->secret('What is the password?');

确认类型输入if ($this->confirm('Do you wish to continue? [y|N]')) { … }

选项$name = $this->anticipate('What is your name?', ['Taylor', 'Dayle']);

索引选项$name = $this->choice('What is your name?', ['Taylor', 'Dayle'], false);

输出绿色信息：$this->info('Display this on the screen');

输入红色信息：$this->error('Something went wrong!');

输入一般信息：$this->line('Display this on the screen');

输出表格：$headers = ['Name', 'Email']; $users = App\User::all(['name', 'email'])->toArray(); $this->table($headers, $users);

进度条：$users = App\User::all(); $bar = $this->output->createProgressBar(count($users)); foreach ($users as $user) {$bar->advance();}$bar->finish();

路由或控制器中触发 Artisan 命令Route::get('/foo', function () {$exitCode = Artisan::call('email:send', ['user' => 1]);});

事件广播：//?lymark <http://laravelacademy.org/post/6851.html>

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

设置缓存为memcached：config/cached中'default' => env('CACHE\_DRIVER', 'memcached')

存到数据库的时候会自动存入缓存，不用作其他处理

缓存：use Illuminate\Support\Facades\Cache; $value = Cache::get('key');

或者传闭包作默认指：$value = Cache::get('key', function() {return DB::table(...)->get();});

缓存项存在Cache::has('key')

缓存项不存在时获取默认值并存入缓存$value = Cache::remember('users', $minutes, function() {return DB::table('users')->get();});第二个参数指存多长时间，永久则是rememberForever

从缓存中获取缓存项然后删除$value = Cache::pull('key');

缓存中存储数据Cache::put('key', 'value', $minutes);

缓存不存在时存储数据Cache::add('key', 'value', $minutes);

永久缓存Cache::forever('key', 'value');

从缓存中删除数据Cache::forget('key'); Cache::flush();

cache辅助函数：取$value = cache('key'); 存cache(['key' => 'value'], $minutes);

缓存标签：Cache::tags(['people', 'artists'])->put('John', $john, $minutes); $john = Cache::tags(['people', 'artists'])->get('John'); Cache::tags(['people', 'authors'])->flush(); Cache::tags('authors')->flush();

自定义缓存驱动//?lymark

将修改保存到缓存中：AppServiceProvider的boot方法中Post::saved(function($post) {$cacheKey = 'post\_'.$post->id;$cacheData = Cache::get($cacheKey);if(!$cacheData){Cache:: add($cacheKey,$post,60\*24\*7);}else{Cache::put($cacheKey,$post,60\*24\*7);}});

安装Gulp： //page

1. 确保安装了node.js $ node -v

2. 安装Gulp： npm install -g gulp

分页: $posts = DB::table('posts')->simplePaginate(3); return view('post.index', ['posts'=>$posts]); 在view中用{!! $posts->render() !!}显示前后页

带页码的分页：paginate(3); 也可以模型：Post::paginate(3);

分页链接到自定义模版:{{ $users->links('partials.paginator') }}

四种日志处理器：single daily syslog errorlog

使用Log记录日志: use Illuminate\Support\Facades\Log; Log::emergency("系统挂掉了",['user\_id'=>1]); 其他alert critical error warning notice info debug

不打印异常：.en中APP\_DEBUG=true

App\Exceptions\Handler中$dontReport中的异常不会记录只会在浏览器中打印

自定义异常显示：abort(403,'对不起，您无权访问该页面！'); 创建resources/views/errors/403.blade.php，{{$exception->getMessage()}}

监听器：

1. EventServiceProvider中'App\Events\PostSaved'=>['App\Listeners\SaveDataToCache']

2. php artisan event:generate

3. app/Events/PostSaved中use Illuminate\Support\Facades\Event; public function \_\_construct(Post $post){$this->post = $post;}

4. app/Listeners/SaveDataToCache中public function handle(PostSaved $event){$post = $event->post; $key = 'post\_'.$post->id; Cache::put($key,$post,60\*24\*7); Log::info('保存文章到缓存成功！',['id'=>$post->id,'title'=>$post->title]); }

5. 触发事件：Event::fire(new PostSaved($post)); //打印到事务中

上传文件：

<<<CREATE <form action="$postUrl" method="POST" enctype="multipart/form-data"> … CREATE;

hasFile $file = $request->file('file'); isValid $destPath = realpath(public\_path ('images')); $filename = $file->getClientOriginalName(); $file->move($destPath, $filename)

Storage:: //page

语言：

1. config/app.php中： 'locale' => 'zh\_cn', 'fallback\_locale' => 'en',

2. resources/lang目录下创建zh\_cn目录，auth.php中'failed' => '用户名/密码不匹配', 。。。

3. 调用: {{$errors->first('email')}} 或@foreach ($errors->all() as $error)<li> {{ trans($error) }}</li> @endforeach

动态本地化语言：

1. Route::get('/{locale}', function ($locale) { App::setLocale($locale); return view('welcome') });

2． resources/lang/en的welcome.php中：<?php return [ 'message' => 'Welcome To Laravel Academy!' ]; resources/lang/zh\_cn的welcome.php中：<?php return ['message' => '欢迎访问Laravel学院！'];

3. resources/views/welcome.blade.php中加入<div class="title">{{ trans ('welcome.message') }}</div>

4. 访问<http://localhost:8000/zh_cn> 或 <http://localhost:8000/en>

邮件

配置：

MAIL\_DRIVER=smtp MAIL\_HOST=smtp.qq.com MAIL\_PORT=465(对应ssl端口,如果是tls则是25) MAIL\_USERNAME= [519264223@qq.com](mailto:519264223@qq.com) MAIL\_PASSWORD=wuopxojkspbhbidi(授权码,通过登录qq邮箱设置) MAIL\_ENCRYPTION=ssl(测试的时候tls不可用)

发送邮件：Mail::send('emails.test',['name'=>$name],function($message){$to = 'lyluoyuan@126.com';$message ->to($to)->subject('测试邮件');});其中emails.test是resources/views/emails/test.blade.php, $ message对应其中的所有内容

发送无需创建视图的纯文本：Mail::raw('这是一封测试邮件', function ($message) {$to = 'lyluoyuan@126.com'; $message ->to($to)->subject('测试邮件'); });

添加附件: $attachment = public\_path('images/兑换物品.png'); $message->attach($attachment,['as'=>'title.png']);

邮件内容中插入图片: 视图中<img src="{{$message->embed($imgPath)}}"> 控制器中: $imgPath = public\_path('images/兑换物品.png'); Mail::send('emails.test',['name'=> $name,'imgPath'=>$imgPath],function($message){ … });

集合转数组：collect([1, 2, 3])->all(); 以某个键的平均值avg('pages') 分割集合chunk(4) 组合集合collapse 键值组合$collection = collect(['name', 'age']); $combined = $collection->combine(['George', 29]);// ['name' => 'George', 'age' => 29] contains('Desk')集合中是否包含'Desk' $collection->contains('product', 'Bookcase')集合'product'键下是否包含'Bookcase'

contain接函数$collection->contains(function ($key, $value) {return $value > 5;});

count diff diffKeys every except(['price', 'discount'])除了指定键的所有集合

通过给定回调过滤集合$filtered = $collection->filter(function ($value, $key) {return $value > 2;}); first flatten多维集合变一维 flatten(1)其中1是深度参数 flip交换键值 forget删除 get groupBy has implode('product', ', ')按键名连接

交集intersect isEmpty keyBy指定键的值作为新集合的键 keys last

遍历map(function ($item, $key) {return $item \* 2;}); mapWithKeys以每个元素为数组作操作 max给定键的最大值 merge min

第n-th个元素创建一个新的集合nth(4, 1);第二个参数是初始偏移

only指定键的集合 按条件分割数组list($underThree, $aboveThree) = $collection->partition(function ($i) {return $i < 3;}); pluck为给定键获取所有集合值，pluck('name', 'product\_id')第二个参数表示如何设置键 pop移除最后面的数据项并返回集合 prepend添加数据项到集合开头，第二个参数为前置项的键 pull通过键从集合中移除并返回剩余数据项 附加数据项到集合结尾push put random(3)

$total = $collection->reduce(function ($carry, $item) {return $carry + $item;},4); 减少集合到单个值,其中$carry为初始4,默认null

使用给定回调过滤集合$filtered = $collection->reject(function ($value, $key) {return $value > 2;}); reverse search按值搜索,返回键 $collection->search('4', true);第二个参数为true表示严格比较,即类型一致。或接函数search(function ($item, $key) {return $item > 5;});

移除并返回第一个数据项shift shuffle打乱 slice(4, 2)类似分页,1参是offset,2参是limit

sort按大小排序 sortBy按指定键排序，或函数sortBy(function ($product, $key) sortByDesc类似sortBy，只是反序

splice(2)从指定位置切开,原集合会改变，第三个参数是被替换的项，加到原集合

split(3)其中3表示分成3部分 sum按键，按函数 take(n)取前几位,负数表示取后几位

toArray toJson transform(function ($item, $key)遍历替换,会改变原集合

union并集 unique,也可以接键或函数unique(function ($item) values where('price', 100)按指定键值对过滤集合, whereStrict为严格过滤 whereIn('price', [150, 200])严格whereInStrict 按集合合并成数组集合zip([100, 200])

高阶消息传递$users->each->markAsVip();

错误/日志<http://laravelacademy.org/post/6869.html> //?lymark

事件: <http://laravelacademy.org/post/6877.html> //?lymark

接本地url：asset(…)

文件系统/存储: Storage::disk('local')->put('file.txt', 'Contents');存入了storage/app/file.txt文件中

获取Storage::get('file.txt'); exists url size lastModified

存Storage::put('file.txt', '$contents'); 头添加Storage::prepend('file.txt', 'Prepended Text');

尾添加Storage::append('file.txt, 'Appended Text');

复制: Storage::copy('old/file1.jpg', 'new/file1.jpg'); 移动move

删除：Storage::delete(['file1.jpg', 'file2.jpg']);

文件数组files allFiles包含子目录 目录directories 包含子目录allDirectories

创建目录makeDirectory 删除目录deleteDirectory

辅助函数

添加给定键值对到数组$array = array\_add(['name' => 'Desk'], 'price', 100);

合并数组$array = array\_collapse([[1, 2, 3], [4, 5, 6], [7, 8, 9]]);

分别返回键值的数组list($keys, $values) = array\_divide(['name' => 'Desk']);

使用”.”号将将多维数组转化为一维数组$array = array\_dot(['foo' => ['bar' => 'baz']]);// ['foo.bar' => 'baz'];

从数组中移除给定键值对$array = array\_except($array, ['price']);

第一个元素$value = array\_first($array, $callback, $default);

多维数组转化为一维数组$array = array\_flatten($array);

使用”.”号从嵌套数组中移除给定键值对array\_forget($array, 'products.desk');

使用”.”号从嵌套数组中获取值$value = array\_get($array, 'names.john', 'default');

使用“.”检查给定数据项是否在数组中存在$hasItems = array\_has($array, ['product.price', 'product.discount']);

array\_last array\_only array\_prepend

返回给定键对应的值列表$array = [ ['developer' => ['name' => 'Taylor']], ['developer' => ['name' => 'Abigail']]]; $array = array\_pluck($array, 'developer.name');// [1 => 'Taylor', 2 => 'Abigail'];

返回移除对应键值对后的结果array\_pull($array, 'name');

在嵌套数组中使用”.”号设置值array\_set($array, 'products.desk.price', 200);

排序$array = array\_sort($array,function($value){return $value['name'];});

递归排序array\_sort\_recursive

对数组进行过滤$array = array\_where($array, function ($value, $key) {return is\_string($value);});

数组的第一个元素head last

绝对路径app\_path 根路径base\_path 配置路径config\_path 数据库配置路径database\_path public路径public\_path resources路径resource\_path storage

转驼峰$camel = camel\_case('foo\_bar'); 移除命名空间后的类名class\_basename

判断给定字符串是否以给定值结尾ends\_with('This is my name', 'name'); starts\_with

转下划线snake\_case 从头截取字符串str\_limit(str, 7); str\_contains

添加字符串到字符串结尾$string = str\_finish('this/string', '/');

字符串是否与给定模式匹配,\*是通配符,不是正则$value = str\_is('foo\*', 'foobar');

字符串转复数str\_plural('child');第二个参数取1则是单数形式 str\_singular转单数

转化为单词开头字母大写的格式studly\_case 字符串转化为Title形式title\_case

给指定控制器生成url：$url = action('HomeController@index');可由第二个参数传参

补充域名成为url：asset https形式：secure\_asset 生成完整路由url('user/profile',[ 'p1']);

url()->previous(); url()->current(); url()->full();

抛出异常abort(401, 'Unauthorized.'); 表达式为true时抛出异常abort\_if(表达式, 403);

abort\_unless

函数返回一个认证器实例auth()->user()

用Bcrypt对给定值进行哈希bcrypt('my-secret-password')

获取配置变量的值config('app.timezone'); 改值config(['app.debug' => true]);

生成一个包含CSRF令牌值的HTML隐藏域{!! csrf\_field() !!}

获取当前csrf令牌的值csrf\_token()

输出给定变量值并终止脚本执行dd, 不终止脚本dump

推送一个新的任务到 Laravel 任务队列dispatch(new App\Jobs\SendEmails);

获取环境变量的值，不存在时返回默认值env('APP\_ENV', 'production');

分发给定事件到对应监听器event(new UserRegistered($user));

获取一次性存放在Session中的值old('value', 'default');

回到重定向的页面return redirect('/home');//跳转到首页return redirect('/');

创建一个响应实例：return response('Hello World', 200, $headers); return response()->json(['foo' => 'bar'], 200, $headers);

使用数据库存储Session: .env中SESSION\_DRIVER= database config/session.php中'connection' => 'mysql'

其他存储memcached redis

session用法$value = session('key'); session(['chairs' => 7, 'instruments' => 3]); $value = session()->get('key'); session()->put('key', $value); has判断存在 添加多个数据到session数组：session()->push('site.xxx', 'http://LaravelAcademy.org'); pull获取数据后删除，也可以用forget flush删除全部 flash仅下一次有效的session reflash保持flash修饰的session keep(['message']);指定哪些session有效

获取一个视图实例view

邮件//?lymark <http://laravelacademy.org/post/6899.html> 通知//?lymark

连接多个数据库DB::connection //需要在config/database.php里设置'test\_ly' => ['database' => 'test\_ly']

命名绑定DB::select('select \* from users where id = :id', ['id' => 1])

填充器：php artisan make:seeder UserTableSeeder 在app/database/seeds中的DatabaseSeeder的run方法中数据库操作 再php artisan db:seed

redis: composer require predis/predis 1.0.\*

基本使用：use Illuminate\Support\Facades\Redis; Redis::set($key,$user->name); Redis::get($key)

队列任务//page

1. 启用redis：.en中QUEUE\_DRIVER=redis

2. php artisan make:job SendReminderEmail 生成app/Jobs/ SendReminderEmail

…

修改器：在模型文件中public function getFirstNameAttribute($value){return ucfirst($value);} first\_name是一个字段名。调用：$user = App\User::find(1); $firstName = $user->first\_name;

set方法：public function setFirstNameAttribute($value){$this->attributes['first\_name'] = strtolower($value);} 调用：$user->first\_name = 'Sally';

类型转换：模型文件中protected $casts = ['first\_name' => 'integer',]; 调用var\_dump($user->first\_name);

隐藏某个属性protected $hidden = ['first\_name'];

显示白名单protected $visible = ['first\_name', 'last\_name'];

临时显示隐藏的属性$user->makeVisible('first\_name')

运行测试：终端phpunit <http://laravelacademy.org/post/7043.html>

会话／认证：public function testApplication(){$response = $this->withSession(['foo' => 'bar'])->get('/');}

模型工厂生成并认证：$user = factory(User::class)->create(); $response = $this->actingAs($user)->withSession(['foo' => 'bar'])->get('/');

测试api响应：$response = $this->json('POST', '/user', ['name' => 'Sally']); $response-> assertStatus(200)->assertJson(['created' => true,]);

验证完全匹配：assertExactJson

浏览器测试（Laravel Dusk）：<http://laravelacademy.org/post/7047.html#toc_22>

AppServiceProvider的register方法中注册该提供者if ($this->app->environment('local', 'testing')) {$this->app->register(DuskServiceProvider::class);}

运行测试php artisan dusk

在DuskTestCase中默认chrome测试：startChromeDriver

5.2新特性-表单数组输入验证功能实现//page

$validator = Validator::make($this->request->all(), [ … ]); if ($validator->fails()) {return $this->displayError($validator->errors()->first());}

隐式路由模型绑定 访问频率限制

操作json属性的数据（MySQL 5.7以上版本）： migration中$table->json('meta'); 查询：$redLovers = DB::table('contacts')->where('meta->favorite\_color', 'red')->get(); 改：DB::table('contacts') ->where('id', 1)->update(['meta->wants\_newsletter' => false]);

Laravel Echo //page ? elixir

$loop变量：@foreach ($pages as $page) …{{$loop->index}}…{{ $loop->count }} @endforeach 如果有嵌套循环：{{ $loop->parent->index }}

图片尺寸验证：$this->validate($request, ['avatar' => 'dimensions:min\_width=250, min\_height=500']); 视图中：<input type="file" name="avatar">

自定义路由文件：参考RouteServiceProvider

全文搜索//page ?//lymark

高阶消息传递：(对集合链式操作) $employees->reject->retired->each->sendPayment();

中间件组：app/Http/Kernel.php中protected $middlewareGroups = ['web' => … , 'api' => … , 'admin' => ['web','auth',]] 使用：Route::group(['middleware' => 'admin'], function () {Route::get('dashboard', function () {return view('dashboard');});});

订阅支付Laravel Cashier：<http://laravelacademy.org/post/7067.html>

vim /Users/jinxiaoli/.composer/composer.json composer update

Envoy

Scout <http://laravelacademy.org/post/7129.html> 自动同步更新模型记录的索引

用户认证5.1:

1. php artisan migrate根据原始migration建表

2. 定义路由：// 认证路由...Route::get('auth/login', 'Auth\AuthController@getLogin');

Route::post('auth/login', 'Auth\AuthController@postLogin'); Route::get('auth/logout', 'Auth\AuthController@getLogout'); // 注册路由...Route::get('auth/register', 'Auth\AuthController@getRegister');Route::post('auth/register','Auth\AuthController@postRegister');

3．创建注册视图resources/views/auth/register.blade.php

4．创建登录视图resources/views/auth/login.blade.php

访问: <http://localhost:8000/auth/register>

AuthController中定义认证成功后的跳转链接protected $redirectPath = '/profile';

Route::get('profile','UserController@profile');在UserController 中：public function profile(Request $request){$user = $request->user();echo $user['name'].'登录成功！';}

用户认证密码重制//?lymark <http://laravelacademy.org/post/1290.html>

Socialite实现登录认证//?lymark <http://laravelacademy.org/post/1305.html>

~------------laravel 5.1

相关视频 <http://www.pfinal.cn/>

mac php5升级到php7 <http://www.jianshu.com/p/0456dd3cc78b>

安装homebrew：<https://brew.sh/index_zh-cn.html> /usr/bin/ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)"

安装wget：brew install wget

安装phpunit <http://www.cnblogs.com/IT--Loding/p/6222147.html>

添加子域名: 即添加域名解析

composer dump-autoload

在别的终端打不开autoload报错的解决： composer update --no-scripts

telnet 192.168.8.102 33060连其他电脑数据库。 连不上：看.en中是否有其他DB连接也要加，或者先试试终端连接mysql -h 192.168.8.102 -P33060 没装msql：brew install mysql

phpstorm激活server <http://idea.ibdyr.com>

安装vagrant：下载地址<https://www.vagrantup.com/downloads.html>

vagrant up报private key错： ssh-keygen -t rsa -b 4096 -C [lyluoyuan@126.com](mailto:lyluoyuan@126.com)

vagrant up需要终端翻墙，用lantern，在lantern的localhost网页页面，查看 设置->高级设置->HTTP(S)代理服务器，记住这个端口号port。 brew install privoxy vim /usr/local/etc/privoxy/config 加入listen-address 0.0.0.0:8118 forward-socks5 / 127.0.0.1:port . sudo /usr/local/sbin/privoxy /usr/local/etc/privoxy/config export http\_proxy='http://127.0.0.1:port export https\_proxy='http://127.0.0.1:port'

报错collides with a non-hostonly network：换个host，可能跟路由有冲突

laravel中的admin包 <https://github.com/z-song/laravel-admin> 路由相关的关联在providers中的 AdminServiceProvider

laravel中的ide生成 <https://github.com/barryvdh/laravel-ide-helper>

解决git冲突：暂存：git stash再git pull

route函数用法 Route::get('show','HomeController@show')->name('show'); redirect()->route('show',['article'=>$article]); protected function show(Request $request){…} 不走post, $request只能传基本类型

在blade视图中显示图片就是src="images/image.jpg"，不用加public\_path，默认

phpMyAdmin不支持JSON?

laravel-admin:

route()用不了试试url(../) 或用asset 多图时删除图片：注释掉app/Http/Kernel.php中的ConvertEmptyStringsToNull

laravel默认加密方式：$password = Hash::make('password');直接粘到数据库就行，即使加密后的password经常变，remember\_token会自动生成

报错data too long：将char类型字段改为Text，注意有可能要改的是admin\_operation\_log这类关联库

ajax: php中json\_encode；js中JSON.parse，数组个数是length

homestead初始数据库用户名密码mysql -uroot –psecret 修改update user set authentication\_string=password('hnb314159') where user='root'; FLUSH PRIVILEGES;

与虚拟机同步：vagrant ssh git clone … 同步可能有等待时间 访问浏览器(清除缓存)如果不是127.0.0.1后面不要加8000端口,sequel连接数据库端口仍是3306

访问浏览器出现EOF：退出lantern或其他翻墙软件

树状表：表结构中有parent\_id和order，模型中$this->setParentColumn('parent\_id'); $this->setOrderColumn('order'); $this->setTitleColumn('title'); 在controller中显示： $content->body(Category::tree());

统计图表：直接看HomeController的index方法吧

一些查错法：先用get方法在浏览器试

?lymark// model()->with

地图模态框：class=\"btn btn-xs btn-default grid-open-map\" 需要更改或导入的：public/packages app/Admin/Extensions app/Admin/bootstrap.php resources/views/admin

->model()->orderByDesc ->expand(function() {return new Table([], [ … ])})下拉展开

开关：->switch($this->switch)其中'switch' => ['on' => ['value' => 1, 'text' => '启用', 'color' => 'success'],'off' => ['value' => 2, 'text' => '停用', 'color' => 'danger'],]

$filter->useModal()是否折叠筛选按钮

下拉选择$filter->is('community\_status', '状态')->select((Province::all()->pluck ('province\_name', 'province\_id')); between: $filter->between('community\_build\_year', '修建年份');

跳转按钮$actions->prepend('<a class="btn btn-xs btn-primary " href="' . action('\App\Admin\Controllers\CommentController@index', ['id' => 1]) . '"> <i class="fa fa-area-chart"></i> 活动统计</a>');

excel导出：use App\Admin\Extensions\ExcelExporter; $grid->exporter(new ExcelExporter(['id' => 'ID','title' => '标题','content' => '内容'])); <https://github.com/Maatwebsite/Laravel-Excel>

Dingo：

1.在项目根目录的composer.json中"require": {"dingo/api": "1.0.\*@dev"} composer update

2.在config/app.php中'providers' => [Dingo\Api\Provider\LaravelServiceProvider::class]

3. php artisan vendor:publish --provider="Dingo\Api\Provider\LaravelServiceProvider"

这时访问浏览器报错，在.env中：API\_DOMAIN=api.app

4.在routes/api.php中use Dingo\Api\Facade\Route;

5. php artisan vendor:publish --provider="Dingo\Api\Provider\LaravelServiceProvider"

6.在.env中API\_SUBTYPE=ly API\_DEBUG=true

7.改服务器/虚拟机配置

8. 测试：Route::group(['version'=>'v1','namespace'=>'App\Http\Controllers\Api',],function () {Route::get('testApi',function (){return Response::json(['status\_code' => 200]);}); });

ajax访问完整url，只需写完整包含http的url即可

访问浏览器出现No input file specified. 解决：vagrant up --provision

每添加个子域名都要这样做。

若还是不行就vagrant reload或重启计算机，翻墙模式下vagrant up --provision

token: <http://laravelacademy.org/post/7083.html> 生成的token会存到数据库

controller中return Response::json(['status\_code'=>200,'data'=>$comments]);其中$comments是模型数组。ajax回调中：$.each($data.data,function (key,item) { … })

DB::raw取别名用的

$activity->update(['total\_contribute\_quantity' => DB::raw('total\_contribute\_quantity+' . $contribute['contribute\_quantity']) ]);

with(['contribute\_type',function($query){ … }])->groupBy()->select()其中contribute\_type对应ContributeType

font-awesome: 将下载的css和fonts放到同一目录

api认证：

1. composer require laravel/passport

2. config/app.php中Laravel\Passport\PassportServiceProvider::class

3. php artisan migrate

4. php artisan passport:install

5. 添加Laravel\Passport\HasApiTokens trait到用户模型中

6. 在AuthServiceProvider的boot方法中调用Passport::routes方法，use Laravel\Passport\Passport;

7. config/auth.php中'guards' => ['api' => ['driver' => 'passport','provider' => 'api\_test\_users' , ] , ], 'providers' => ['api\_test\_users' => ['driver' => 'eloquent','model' => App\Models\ApiTestUser::class,]],

8 . (测试) $token = $worker->createToken($worker->user\_name . " token")->accessToken; headers['Authorization']='Bearer '.$token

微信开发：验证url和token，不考虑安全的话return $request->input('echostr');

php中发送请求：guzzle： composer require guzzlehttp/guzzle

全局函数：创建app/helpers.php, 写入if( ! function\_exists(‘my\_func’)){function…} composer.json写入"autoload": {"files":["app/helpers.php"]} composer dump-autoload

laravel-admin: 本地能服务器不能上传图片问题：sudo chmod -R 775 上传图片的路径

上传图片报fileinfo的错：没有fileinfo模块。 php –m：查看安装的模块

安装fileinfo: cd /var/lnmp1.3-full/src tar -zxvf php-7.0.7.tar.gz cd php-7.0.7/ext/ fileinfo /usr/local/php/bin/phpize ./configure -with-php-config=/usr/local/php/bin/php-config make && make install vim /usr/local/php/etc/php.ini去掉fileinfo前面的逗号,并添加extension=php\_fileinfo.so /etc/init.d/php-fpm restart 可能会报找不到php\_fileinfo的错：进入make && make install最后生成的文件夹，将fileinfo.so复制为php\_fileinfo.so

安装fileinfo(不好用)：pecl install fileinfo 安装网址 <https://pecl.php.net>

开通https 阿里云下->证书服务->购买证书->选择免费的

七牛使用：上传表单：<form method="post" action="http://up-z2.qiniu.com" enctype="multipart/form-data"> <input name="token" type="hidden" value={{$upToken}}> <input name="file" type="file" /> <input type="submit" value="上传"/></form>

其中获取upToken：$bucket = 空间名(必须); $accessKey = ''; $secretKey = ''; $auth = new Auth($accessKey, $secretKey); $upToken = $auth->uploadToken($bucket);

如果之前用的磁盘存储，然后再改成七牛云存储，服务器上需要重新建立个代码库

登录报错：You must set the encryption key going forward to improve the security of this library。解决：composer.json中"laravel/passport": "^3.0"，然后composer update