Composer

composer 是PHP包管理器

composer 使用

下载:
https://getcomposer.org/download/
中文镜像:
composer config -g repo.packagist composer https://packagist.phpcomposer.com
安装:
composer init 初始化
composer require <包名>

laravel 下载

`composer create-project --prefer-dist laravel/laravel blog`

搭建本地虚拟域名

`参考 本地搭建虚拟域名.txt`

配置

程序秘钥
php artisan key:generate
修改时区
config/app
timezong => 'PRC';
打印
dump()
dd()
获取和设置配置
Config::get('app.timezone');
Config::set('app.locale', 'cn');
关闭和重启应用
关闭 php artisan down
resources\views\errors\503.blade.php
启动 php artisan up
resources\views\welcome.blade.php

Laravel 路由

```
基本路由
    Route::get('url 匹配服务器绝地地址 域名后面字符串','callback');
    Route::post('url 匹配服务器绝地地址 域名后面字符串','callback');
    Route::get($uri, $callback);
    Route::post($uri, $callback);
    Route::put($uri, $callback);
    Route::patch($uri, $callback);
    Route::delete($uri, $callback);
带参数的路由 并且 限定
    Route::get('admin/{id}',function($id){
        echo $id;
    })->where('id','[0-9]+');

路由别名
    //带别名的路由(一)
    Route::get('admin/goods/delete',['as'=>'agd','uses'=>function(){
        dump('商品删除');
    }]);
    //带别名的路由(二)
    Route::get('admin/goods/delete2/{id}',function(){
        dump('商品删除22222222');
        route(); //通过别名创建url
        url(); //通过字符串创建url
        redirect(); //跳转
    })->name('gd');

404页面设置
    设置模板 -> resources/views/errors/404.blade.php

csrf 保护
    1. form -- post
        {{ csrf_field() }}

    2. ajax -- post

        - header
            <meta name="csrf-token" content="{{ csrf_token() }}">
        - ajax
            $.ajaxSetup({
                headers: {
                    'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
                }
            });
```

中间件 筛选路由 帮组执行各式各样的任务

1. 创建 php artisan make:middleware LoginMiddleware

2. 填写实例代码:
```
// 验证登录
if (session('admin_login')) {
    // 执行下一次请求 通过
    return $next($request);
}else{
    // 跳转到登录页面
    return redirect()->route('login');
}
```
3. 注册中间件
```
app\http\Kernel.php
    'login'=> \App\Http\Middleware\LoginMiddleware::class,
```
4. 使用
```
Route::get('/', function () {
    // 加载视图
    return view('welcome');
})->middleware('login');

// 使用 路由组对一组路由进行统一管理
Route::group(['middleware'=>'login'],function(){

})
```

控制器

```
创建
    php artisan make:controller                普通控制器
    php artisan make:controller --resource     资源控制器

访问:
    Route::get('home/goods/index','GoodsController@index');
    Route::get('home/goods/show/{id}','GoodsController@show');

    Route::resource('photos','PhotosController');
    资源控制器操作处理#
    动作          URI                  操作
    GET         /photos               index
    GET         /photos/create         create
    POST        /photos               store
    GET         /photos/{photo}        show
    GET         /photos/{photo}/edit    edit
    PUT/PATCH   /photos/{photo}        update
    DELETE      /photos/{photo}        destroy
```

Request 请求

基本信息获取

```
function index(Request $request)
```

```
{

}
请求路径
    $request->path()
检查路径
    $request->is('admin/*')
获取完整url
    $request -> url()
获取请求的方式
    $request->method();
检测当前请求的方式
    $request->isMethod('post')
检测字段是否存在
    $request->has('name')
检测字段 的值 是否为空
    $request->filled('name')
获取请求参数
    $name = $request->input('name');
默认值设置 获取一个参数
    $request->input('name', 'Sally');
获取所有参数
    $request -> all()
获取指定
    $request->only(['username', 'password']);
去除当前指定
    $request->except(['credit_card']);
```

闪存信息

```
将所有的请求参数写入闪存中
    $request->flash()
    案例：用户名：<input type="text" name='username' value="{{ old('username') }}">
将部分参数写入闪存中
    $request->flashOnly('title','price')
将某些值去除后存到闪存中
    $request->flashExcept('_token');
简便使用
    return back()->withInput();
```

cookies && session

```
cookie
    设置
        \Cookie::queue('name','iloveyou',10);
            设置cookie的时间为10分钟
        return response('haha')->withCookie('uid',10,10);
    读取
        $request->cookie('name');
        \Cookie::get('name');
```

```
session
    设置：  session(['key'=>'value'])
    获取：  session('key')

    session(['a'=>'v'])
    session('a')
```

文件操作 -- 上传

```
检查是否有文件上传
    if ($request->hasFile('photo')) {
        //
    }
文件上传
    $file = $request->file('profile');//创建文件上传对象
    $file->store('public');//上传到指定的文件夹

    // 获取文件后缀
    $ext = $file->extension();
    // 拼接名称
    $file_name = time()+rand(1000,9999).'.'.$ext;
    $res = $file->storeAs('public',$file_name);

读取文件
    ------------------------------------
        创建软连接 php artisan storage:link
        注意: $file->store('public');
    ------------------------------------
        直接修改配置文件 config/filesystems.php
            'local' => [
                'driver' => 'local',
                // 'root' => storage_path('app'),
                'root' => public_path('uploads'),
            ],
```

响应对象

```
显示模板
    return view('goods.add');
返回json
    return response()->json(['a'=>100,'b'=>2000]);
    echo json_encode();

下载文件
    return response()->download('文件的url地址','文件名称');
页面跳转
    return redirect('/goods/add'); //跳转到指定路由
    return redirect()->route('profile', ['id' => 1]); //跳转到指定的别名
    return redirect()->action('HomeController@index');    //跳转到指定的控制器中方法
```

视图

```
分配数据到模板
    view('user.edit',['username'=>'admin'])
获取数据
    {{ $name }}
解析html标签
    {!! $str !!}
默认值设置
    {{ $str2 or 'default value' }}
注释
    {{--  注释 特点:不解析任何变量  --}}
使用函数
    {{ time() }} {{ md5() }}
流程控制
    @if($a == 1)

    @elseif(*)

    @else

    @endif
三元运算符
    {{ $a == 1 ? 'true' : 'false' }}
for循环
    @for($i = 0;$i<10;$i++)

    @endfor

foreach 遍历
    @foreach($data as $K=>$V)
    @endforeach

模板引入
    @include()
模板继承
    @extends('')

模板占位符
    @yield('title')
    @section()
    @show
模板填充
    @section('')
    @endsection
```

数据库

注意:数据库使用 编码 `utf8mb4`

链接数据

```
配置 `.env` 文件
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lamp_oto
DB_USERNAME=root
DB_PASSWORD=
```

基本操作

```
查询
   DB::select('select * from users where active = ?', [1]);
插入
   DB::insert('insert into users (id, name) values (?, ?)', [1, 'Dayle']);
修改
   DB::update('update users set votes = 100 where name = ?', ['John']);
删除
   DB::delete('delete from users');
```

查询构造器

查询

```
查询所有  DB::table('users')->get()
查询单条  DB::table('users')->first()
查询字段  DB::table('users')->select('name', 'email as user_email')->get()
```

插入

```
单条 bool
   DB::table('users')->insert(
      ['email' => 'john@example.com', 'votes' => 0]
   );
多条 bool
   DB::table('users')->insert([
      ['email' => 'taylor@example.com', 'votes' => 0],
      ['email' => 'dayle@example.com', 'votes' => 0]
   ]);
获取ID插入 插入数据返回最后的id号  只能插入的单条
   $id = DB::table('users')->insertGetId(
      ['email' => 'john@example.com', 'votes' => 0]
   );
```

修改

```
返回的是受影响的行数
DB::table('users')
        ->where('id', 1)
        ->update(['votes' => 1]);
```

删除

```
返回的是受影响的行数
DB::table('users')->where('id', '<', 100)->delete();
```

连贯操作

```
条件
    DB::table('users')
            ->where('name', 'like', 'T%')
            ->get()
    DB::table('users')
                ->where('votes', '>', 100)
                ->orWhere('name', 'John')
                ->get();
    DB::table('users')
                ->whereBetween('votes', [1, 100])->get()
    DB::table('users')
                ->whereIn('id', [1, 2, 3])
                ->get()

    计算
        总和
            DB::table('users')->count()
        最大值
            DB::table('orders')->max('price')
        最小值
            DB::table('orders')->min('price')
        平均值
            DB::table('orders')->avg('price')
    排序
        orderBy('name', 'desc')

    表链接
        DB::table('users')
            ->join('contacts', 'users.id', '=', 'contacts.user_id')
            ->join('orders', 'users.id', '=', 'orders.user_id')
            ->select('users.*', 'contacts.phone', 'orders.price')
            ->get();
```

事物处理

注意数据表引擎 `innodb`

开启事务　DB::beginTransaction();
提交事务　DB::commit()
回滚事务　DB::rollBack()

自定义全局函数

1. 在app中创建文件夹 App\common\function.php
2. 在composer.json 添加files选项

```
"autoload": {
      "classmap": [
         "database"
      ],
      "files":[
         "App/common/function.php"
      ]
   },
```

3. 更新composer

```
composer dump-auto
```

sql 语句打印

debugbar安装
添加　composer require barryvdh/laravel-debugbar
卸载　composer remove barryvdh/laravel-debugbar

如何去学习框架

1. 安装 环境
2. 文件目录 结构 MVC
3. 学习框架中url　路由
4. 控制器
   - 创建
   - 请求
   - 响应
5. 模板 视图
   - 加载
   - 模板语法
6. 模型 数据库
   - 增删查改
7. 一个管理模块
8. 搜索 分页 验证码 文件上传