**一 Views**

分离控制器和呈现   
1) 存储位置：resrouces/views/ config/view.php指定了paths，默认的模板目录   
2) 呈现方式：return view('<viewName>',[]);   
return view()->make('<viewName>',[]);   
a) 第一个参数：视图文件的名称：   
子文件模式： .语法； /语法   
两种视图文件名称的区别：   
1) viewName.blade.php   
laravel通用，可以使用所有的@语法，   
2) viewName.php   
普通的模板，只能使用原生的

b) 第二个参数：传递数据到视图

三种方式：

1) 数组方式： view('',[ 'key1' => $val1, 'key2' => $val2 ])

//view('',$array()) 视图中用@foreach

2) with方式：

传统方法:

view('<viewName>')->with('key1',$val1)

//with(['key1'=>$val1, 'key2'=>$val2])即with(array())

魔术方法:

view('<viewName>')->withKey1($val1)->withKey2($val2)->...;

视图中 {{ $key }}

3) compact方式

compact('key1','key2')

//视图中foreach

视图解析方式：//后边解释两种方式的区别

{{ $key1 }} {!! $key1 !!}

3) 查看视图是否存在   
if(view()->exists('<viewName>') { }   
view方法不带参数返回的是Illuminate\Contracts\View\Factory的一个实例，可以调用其中的任何方法

4) 与所有视图共享数据   
在AppServiceProvider中的boot方法中使用：   
view() -> share($key,$value)   
// view() -> share(['key1'=>$val1, 'key2'=>$val2])   
// 在视图中直接使用：{{ $key1 }} {{ $key2 }}

5) 视图组件 View Composers   
视图被渲染前，所调用的闭包或者类方法   
ComposerServiceProvider的boot方法中使用   
想在每次渲染某些视图时绑定数据//什么情况下用   
If you have data that you want to be bound to a view each time that view is rendered   
在Service Provider中注册组件   
a) 定义一个视图组件

1在service Provider中定义：

2定义方法：view()->composer('<viewName>','App\...\DiyComposer');//指定自己定义的composer

//view()->composer('<viewName>',function($view) { $view->with('key', $val) ;} )

需要在config/app.php中定义该视图组件(providers数组中添加)

b) 创建DiyComposer

public function composer(View $view) { $view->with('key',$val) ;}

Now that we have registered the composer, the DiyComposer@compose method will be executed each time the <viewName> view is being rendered

//为什么每次会自动调用？？？   
c) 绑定多个view，可以用通配符:   
view()->composer('\*',...);   
//view()->composer(['view1','view2'],...);   
d) 扩展视图   
在ServiceProvider中定义$this->loadsViewsForm   
protected function loadViewsFrom($path, $namespace) {   
if (is\_dir($appPath = $this->app->basePath().'/resources/views/vendor/'.$namespace)) {   
$this->app['view']->addNamespace($namespace, $appPath);   
}   
$this->app['view']->addNamespace($namespace, $path);   
}   
先找views/vendor/$namespace下是否有模板文件存在，没有则去指定的$path中寻找   
//使用publishes方法发布视图 ： vendor:publish 视图目录将会被复制到所特定的目录   
6） View Creators   
同View Composer 但视图渲染后将立即执行

二 Blade Templates   
Blade 是 Laravel 所提供的一个模板引擎。   
两大优点:template inheritance | sections   
blade模板以.blade.php后缀； -----> 指令集   
1) Template Inheritance   
每个页面是由多个blade模板组合而成的   
@extends :   
每个模板只能使用一次@extends指令：即某个模板只能继承于某一个父模板   
但可以实现每个模板之间都存在继承关系：a继承于b，b继承于c   
@yield   
a)定义页面布局

指定基本的html页面结构   
指定@yield等 可以指定第二个参数作为默认值   
b)视图模板中继承上面定义的布局模板 @extends   
@section/@endsection 中实现@yield中内容的填充   
如果是特别简单的字符串赋值，可以单纯的使用@section('sectionName','value')即可；复杂的用@section / @endsection填充   
可以使用@section('')/@show 来定义区块   
如果在子模板中想显示父模板中的section，可以使用@section('') @parent @endsection   
如果子模板想覆盖父模板的section ，可以使用@section/@endsection或者用@section/@overwrite   
@include   
@include('sub-veiw',array()):第二个参数可以向子页传递数据

2) 在blade模板中显示数据   
{{ }} {!! !!} //{{{ }}}   
两者的区别：   
a){{ }}对HTMl实体字符进行转义 <>防止XSS攻击？？html标签直接输出   
b){!! !!}数据不被转义 //html标签被浏览器解析   
// vendor/laravel/framework/src/Illuminate/View/compilers/BladeCompiler.php   
{{ }}中可以使用一些简单的php语法 :   
？ ：(三元运算） // or等 检测数据是否存在

{{ 及 {!!前加@可以跳过blade模板引擎的解析直接输出 如：@{{$test}} : 直接原样输出

在输出由用户提供的内容时候需要非常小心，使用{{ }}转义内容中的实体字符串

3) @if @endif @else等指令

a) @if @elseif @else @endif

b) @unless(condition) @endunless

//除非condition成立，不然执行之间的语句

c) Loops

1) @for @endfor

2) @foreach( as ) @endforeach

3) @while @endwhile

4) @forelse( as ) @empty @endforelse

5) @each('viewName',array(),'eachVariableName'[,arrayEmptyRenderedThisView])

4) 注释

{{-- --}}

5) 引入服务

@inject('serviceName','class\interface')

6) 自定义指令 ?????

使用directive方法，在ServiceProvider的boot方法中注册

Blade::directive('',function(){});

三 Form //FormBuilder.php

Form引入方法:（添加了Html和Form)

1) composer require illuminate/html

2) composer.json中 添加:

"require" { "illuminate/html":"~5.0"} -> composer update

注册Html服务：config/app.php providers数组中添加,并在aliases数组中设置别名Form Html

模板中使用方法: {!! !!} //{{ }} 会原样输出

1) {!! Form::open() !!} / {!! Form::close() !!}

open的参数：以键值对存在的数组

a) method/ url/ route/ action/ files

method：默认post,get/\\\

delete/ put方法会自动添加隐藏的\_method字段

action: 指定Controller@Method

url : 指定Route中的路径

files : 是否允许上传文件

b) 可以自己添加其他属性，如class等

c) 自动在session中存储随机token，在表单中添加隐藏的\_token字段，以防跨网站请求伪造攻击

d) 附加CSRF过滤器到路由

Route::post('',array('before' => 'csrf' ,functioin() { });

2) Form Model Binding 将模型内容添加至表单中

Form::model($model,array())//替换Form::open

public function model($model, array $options = array()) {

$this->model = $model;

return $this->open($options);

}

模型值自动比对字段名并填充

session中的数据 >> 用户传递的数据 >>模型属性数据

3) 用法

a) Form::label($name,$value,array())

b) Form::input($type,$name,$value,array())

c) Form::text($name,$value,array())

d) Form::textarea($name,$value,array())

e) Form::password($name,array());

f) Form::hidden($name,$value,array());

g) Form::email($name,$value,array());

h) Form::url($name,$value,array());

i) Form::file($name,array());

j) Form::checkbox($name,$value =1,$isChecked,array());

k) Form::radio($name,$value,$isChecked,array() );

l) Form::select($name,$list=array(),$selected =null, array());

l) Form::reset($value,array())

4) 自定义表单宏

Form::macro('<filedName>',function() {return ...})

Form::<fieldName>即可

5) Html用法 //HtmlBuilder.php

a) {!! Html::style('url',$attributest(),$secure[http|https]) !!}

b) {!! Html::script('url',$attributest(),$secure[http|https]) !!}

c) image(<img src...) / link(<a href...)[secureLink] /

listing($type,$list,$arrributes()) // ul/ol

四 Theme   
<https://github.com/teepluss/laravel-theme>安装教程   
1) composer.json : 'teepluss/theme' : 'dev-master'   
2) config/app.php中注册Theme服务   
之后php artisan 多出了theme命令   
theme:create   
theme:destroy   
theme:widget   
3) 执行 $ php artisan vendor:publish --provider="Teepluss\Theme\ThemeServiceProvider"   
发布包进行发布？？？不知道为什么要发布   
when users of your package execute Laravel's vendor:publish Artisan command, your package's views will be copied to the specified location.   
Copied File [/vendor/teepluss/theme/config/theme.php] To [/config/theme.php]   
Publishing complete for tag []!

intentor   
1) composer引入   
2) service注册   
3) vendor:publish   
...

4) Creating

$ php artisan theme:create themeName

direc: public/themes/themeName

//删除 $php artisan theme:destroy

// 代码创建 Artisan::call('theme:create', ['name' => 'foo', '--type' => 'blade']);

5) 配置 Configuration config/theme.php

方便配置css/js, 面包屑模板 / metas

6) usage

1) 在config/app.php配置主题名称（或当前主题所在目录）

'theme' => 'default';

// Config::get('app.theme')获取当前主题名称

// : return view($tpl)

ThemeServiceProvider要在ViewServiceProvider后注册

setAlias:"Theme"

Themes Structure:   
themes   
|\_\_mytheme   
|\_css/   
|\_less/   
|\_js/   
|\_views/   
|\_info.php   
要覆盖默认的view，则在views中创建相应的文件