**暂时不看，连已有的东西都没学会，等做出两个项目再说**

**通过 View Composer 预设视图组件数据变量**

我们已经在[视图使用](https://laravelacademy.org/post/9620.html)这篇教程演示了如何从后端传递数据给视图模板，但是这里有个场景需要拉出来讨论，我们的视图有很多公共部分，比如导航菜单、侧边栏、底部信息等，通常我们会以单独的视图组件来处理这些元素区块，但是如何从后端传递这些组件需要的数据变量是个问题，因为这些组件在多个页面中共用，从后端角度来看，会涉及到多个路由/控制器方法，难道我们要每次都重复获取并传递这些数据吗？有没有一种方式可以支持一处定义，多处复用？

答案是有，在 Laravel 中，我们可以通过 View Composer 功能来实现上述需求，我们可以在后端通过 View Composer 将数据绑定到指定视图，从而避免在路由定义或控制器方法中重复获取以及显式传递这些视图组件所需的数据。

废话不多说，接下来我们就来演示 View Composer 的使用，假设我们有一个侧边栏视图组件 resources/views/partials/sidebar.blade.php 用于显示网站最新发布的五篇文章，该组件会在每个视图中引入，如果不使用 View Composer 的话，需要在每个路由定义（或者控制器方法）中这么传递数据：

Route::get('home', function () {

return view('home')->with('posts', Post::recent());

});

Route::get('about', function () {

return view('about')->with('posts', Post::recent());

});

这些数据变量最终会在每个视图中通过引入 partials.sidebar 组件传递到侧边栏。这样的写法两三个还能忍，十个八个的话就让人抓狂了，好在我们还可以全局「预设」这些视图变量，通常这个工作需要在某个服务提供者的 boot 方法中进行，现在我们将其定义到 app/Providers/AppServiceProvider.php 的 boot 方法：

view()->share('posts', Post::recent());

如果不指定视图组件的话，上述代码的含义是在所有视图中共享 posts 变量（该用法在[视图入门教程](https://laravelacademy.org/post/9620.html#toc_2)中已经提及），这当然是有点浪费了，不推荐这么做，我们通常会以闭包方式通过 View Composer 指定视图作用域来预设共享「变量」：

view()->composer('partials.sidebar', function ($view) {

$view->with('posts', Post::recent());

});

这样，我们就可以在 resources/views/partials/sidebar.blade.php 中使用 posts 变量，而不必在定义路由或实现控制器方法的时候显式传递它了。

你甚至还可以通过数组/通配符的方式指定多个视图作用域：

// 通过数组指定多个视图组件

view()->composer(['partials.header', 'partials.footer'], function ($view) {

$view->with('posts', Post::recent());

});

// 通过通配符指定多个视图组件

view()->composer('partials.\*', function ($view) {

$view->with('posts', Post::recent());

});

**通过自定义类实现更加灵活的数据预设**

除了常见的闭包方式外，你还可以通过自定义类的方式为 View Composer 实现更加灵活的数据预设。

首先，我们在 app/Http/ViewComposers 目录下创建一个自定义 View Composer 类 RecentPostsComposer.php：

<?php

namespace App\Http\ViewComposers;

use App\Post;

use Illuminate\Contracts\View\View;

class RecentPostsComposer

{

private $posts;

public function \_\_construct(Post $posts) {

$this->posts = $posts;

}

public function compose(View $view) {

$view->with('posts', $this->posts->recent());

}

}

我们在 RecentPostsComposer 类的构造函数中注入了一个 Post 模型类，该模型类会在实例化的时候自动注入，然后我们将变量预设逻辑定义在 compose 方法中。这样，当我们在 View Composer 中调用 RecentPostsComposer 类的时候，compose 方法会被自动调用从而完成变量预设：

view()->composer( 'partials.sidebar', \App\Http\ViewComposers\RecentPostsComposer::class );

我们可以通过这段代码取代之前的闭包函数定义的 View Composer，但是除非预设逻辑很复杂，否则推荐使用闭包函数方式来实现，一则简洁，二则减少了不必要的类初始化和方法调用对性能的损耗。

**在视图中注入服务**

我们在 [Blade 模板引擎入门教程](https://laravelacademy.org/post/9662.html)中演示了如何在视图模板中处理基本变量、集合数据以及对象数据，除此之外，还可以通过服务注入指令 @inject 在视图模板中注入服务，以便快捷使用服务中提供的方法，该功能的初衷和 View Composer 差不多，都是为了避免每次从路由定义/控制器方法中显式重复传递变量到视图模板，提高开发人员的工作效率：

@inject('analytics', 'App\Services\Analytics')

<div class="finances-display">

{{ $analytics->getBalance() }} / {{ $analytics->getBudget() }}

</div>

其原理就是将注册到服务容器中的服务解析出来，然后就可以调用该服务提供的方法：

$analytics = app('App\Services\Analytics');

如果你还不了解服务容器及其工作原理，可以等到后面讲完服务容器后再回来看这个功能，而且在实际生产环境中，学院君不推荐使用这个服务注入功能，因为这很容易将业务逻辑混合到视图模板中，视图层干好数据渲染的事情就好了，数据的处理和获取交由服务端去完成。

**自定义 Blade 指令**

前面我们已经见识过很多基于 Blade 指令实现的功能了，比如[控制结构](https://laravelacademy.org/post/9662.html#toc_2)、[模板继承](https://laravelacademy.org/post/9663.html)、[服务注入](https://laravelacademy.org/post/9664.html#toc_2)等，Blade 指令的强大之处不止于此，还提供了接口让我们可以自定义满足自己特定需求的指令。我们可以通过自定义 Blade 指令替换那些在多处重复编写的、实现同样功能的代码，从而提高代码的可读性和复用性。

比如视图模板中一个很常见的功能就是格式化显示时间，我们可以通过 Blade::directive 方法为其编写一个自定义指令。和 View Composer 一样，需要在 AppServiceProvider 的 boot 方法中注册这个指令：

Blade::directive('datetime', function($expression) {

return "<?php echo ($expression)->format('Y/m/d H:i:s'); ?>";

});

第一个参数是方法名，第二个参数是一个闭包函数，用于定义指定实现逻辑。这样，我们就可以在视图模板中通过 @datetime($time) 指令统一显示指定格式的日期时间了。

注：更新完 Blade 指令逻辑后，必须删除所有的 Blade 缓存视图指令才能生效。缓存的 Blade 视图可以通过 Artisan 命令 view:clear 移除。

除此之外，我们还可以通过 Blade::if 方法在 Blade 模板中实现自定义的 if 指令，具体实现方式请参考[官方文档](https://laravelacademy.org/post/9550.html#toc_19)。