除了基本的数据渲染及控制结构指令之外，Blade 还提供了模板继承和组件引入功能，从而允许视图模板之间继承、覆盖及引入。

**通过 @yield 和 @section/@show 在布局文件中定义插槽**

在理解 Blade 模板继承的时候，我们可以类比类的继承机制：在父类中定义抽象方法或公共方法，然后在子类中实现抽象方法或重写公共方法。在视图文件中，这个「父类」一般对应布局文件，不同的功能模块往往有不同的页面布局，比如前台、后台、用户中心，页面布局往往不一样。而「子类」则对应不同功能模块的各个子视图页面，比如首页、文章详情页、文章编辑页等等。

我们先来看一个布局文件的示例：

<!-- resources/views/layouts/master.blade.php -->

<html>

<head>

<title>Laravel学院 | @yield('title', '首页')</title>

</head>

<body>

<div class="container">

@yield('content')

</div>

@section('footerScripts')

<script src="{{ asset('js/app.js') }}"></script>

@show

</body>

</html>

在这个布局文件中我们使用了两个 Blade 指令，@yield 用于指定需要子视图继承实现的内容区块，我们可以通过传递第二个参数给该指令用于指定子视图未继承时的默认值，@section/@show 也用于指定子视图需要继承实现的内容区块，并且提供了默认区块内容，与 @yield 不同之处在于，@section/@show 指定的默认内容子视图可以通过 @parent 访问，而 @yield 指定的默认内容对子视图不可见。

**通过 @extends 和 @section/@endsection 在子视图实现继承**

定义好布局文件后，接下来我们来定义继承布局文件的子视图：

<!-- resources/views/dashboard.blade.php -->

@extends('layouts.master')

@section('title', '管理后台')

@section('content')

环境访问 Laravel 学院后台！

@endsection

@section('footerScripts')

@parent

<script src="{{ asset('js/dashboard.js') }}"></script>

@endsection

在子视图中，我们一一实现了布局文件中定义的、需要子视图继承实现的区块内容：

* 首先，通过 @extends 指令指定要继承的布局文件，通过目录名和文件名并以「.」分隔来指定布局文件（Blade 都是通过这种方式指定视图文件，前提是这些视图文件都位于 resources/views 目录中）
* 然后通过 @section 指令依次实现布局文件中需要子视图继承实现的区块内容，两者通过 @section 指令第一个参数建立关联（可以类比为类的继承中的方法名），不同的继承方式实现也略有不同。对于 title 这种比较简单的区块元素我们直接通过传递第二个参数简单实现即可，content 部分是页面主体内容，所以需要通过完整的 @section/@endsection 来实现，最后是 footerScripts 区块，由于布局文件中通过 @section/@show 定义，所以我们可以在子视图中通过 @parent 渲染布局文件中指定的默认区块内容（类比于 PHP 类中通过 parent:: 调用父类方法），并添加该视图中需要的新区块内容。
* 最终子视图页面将是布局文件根据子视图实现填充完所有待继承插槽后呈献给用户。

**通过 @include 引入其他视图组件**

和 PHP 类除了通过单一继承机制外，还可以通过 Trait 横向扩展功能一样，Blade 视图也可以借助 @include 指令引入其他组件完善页面功能，同时这些组件可以在不同视图文件中共用，提高了代码的复用性。比如我们定义一个点击按钮组件：

<!-- resources/views/sign-up-button.blade.php -->

<a class="button button--callout" data-page-name="{{ $pageName }}">

<i class="exclamation-icon"></i> {{ $text }}

</a>

然后就可以在其他视图中通过 @include 引入这个组件：

<!-- resources/views/home.blade.php -->

<div class="content" data-page-name="{{ $pageName }}">

<p>为什么要注册 Laravel 学院: <strong>能提供更多服务</strong></p>

@include('sign-up-button', ['text' => '看看到底有哪些服务'])

</div>

引入组件的时候可以通过传递第二个参数指定组件中需要用到的变量。

注：你也可以不显式指定要传递的参数，组件视图可以访问引入它的视图中的所有变量，但是不推荐这些做，如果被多个视图引入的话容易引起混乱。

**通过 @each 指令循环引入单个组件**

在某些场景下，你可能需要遍历一个集合并循环引入单个组件，这可以通过 @each 指令快速实现。比如我们的侧边栏由多个模块组成（每个模块 DOM 结构一样，可以通过单个组件多次复用实现），我们需要循环引入模块组件，并且为它们设置不同的标题，通过 @each 指令，我们可以这么做：

<!-- resources/views/sidebar.blade.php -->

<div class="sidebar">

@each('partials.module', $modules, 'module', 'partials.empty-module')

</div>

<!-- resources/views/partials/module.blade.php -->

<div class="sidebar-module">

<h1>{{ $module->title }}</h1>

</div>

<!-- resources/views/partials/empty-module.blade.php -->

<div class="sidebar-module">

No modules :(

</div>

@each 指令支持多个参数，第一个参数用于指定要循环引入的组件名，第二个参数是要遍历的集合变量，第三个参数是在引入组件中使用的变量名（对应 $modules 集合中单个元素），最后一个参数是集合数据为空时引入的默认组件。

**通过 @slot 和 @component 实现更加灵活的内容分发**

从 Laravel 5.4 开始，除了通过 @include 引入组件之外，还可以通过 @slot 和 @component 指令在 Blade 中实现更加灵活的内容分发，关于这个功能，应该是借鉴自 Vue.js，Vue 组件中也有[使用插槽分发内容](https://cn.vuejs.org/v2/guide/components.html" \l "%E4%BD%BF%E7%94%A8%E6%8F%92%E6%A7%BD%E5%88%86%E5%8F%91%E5%86%85%E5%AE%B9)的功能。

要在 Blade 中使用插槽分发内容，首先需要创建相应的组件：

<!-- /resources/views/alert.blade.php -->

<div class="alert alert-danger">

{{ $slot }}

</div>

然后在需要引入该组件的地方通过 @component 引入：

@component('alert')

<strong>哎呦!</strong> 出错啦!

@endcomponent

@component 第一个参数对应要引入的组件名，引入组件中 $slot 变量的值通过在引入时 @component 和 @endcomponent 之间的区块内容指定。这种通过插槽分发内容功能的灵活之处在于可以在引入组件的地方定义要渲染的区块内容，换句话说，就是 $slot 的作用域在引入它的父视图中，组件要显示什么内容由引入它的视图决定。

和 @include 一样，@component 也支持传递额外的变量参数到组件中，比如我们修改组件文件如下：

<!-- /resources/views/alert.blade.php -->

<div class="alert alert-danger">

<div class="alert-title">{{ $title }}</div>

{{ $slot }}

</div>

然后就可以在引入它的地方这样传递参数指定 $title 的值：

@component('alert', ['title' => $title])

<strong>哎呦!</strong> 出错啦!

@endcomponent