










LARAVEL FRAMEWORK

BÀI 3: SỬ DỤNG BLADE TEMPLATE TRONG LARAVEL PHẦN 1






- ① Sử dụng Blade để triển khai view
- ① Tạo layout với Blade template



Phần I:

-  Giới thiệu về Blade
-  Hiện dữ liệu trong Blade
-  Các cú pháp hỗ trợ trong Blade
-  Lặp foreach qua 1 mảng trong Blade
-  Cú pháp lặp for trong Blade
-  Viết mã php trong blade
-  Nhúng view con và blade view

Phần II:

-  Tạo layout trong Laravel
-  Tạo layout với template Inheritance
-  Sử dụng các lệnh trong layout
-  Sử dụng stack
-  Sử dụng form



- ❑ Blade là công cụ đã tích hợp sẵn trong Laravel giúp tạo view và layout.
- ❑ Blade rất mạnh mẽ, giúp ích nhiều khi thực hiện view, layout.
- ❑ View trong Laravel có 2 loại : view thường và blade view.
- ❑ Các view dạng blade có tên file mở rộng là **.blade.php**, lưu trong folder resources/views



- ❑ Nạp blade view có thể thực hiện trong routes/web.php, hoặc trong action của controller bằng hàm **view** như view bình thường.
- ❑ Ví dụ:

```
Route::get('/lienhe',function(){  
    return view("lienhe");  
});
```

```
<!--resource/views/lienhe.blade.php-->  
<h1>Liên hệ với chúng tôi</h1>  
<p>Điện thoại: 0198 123 456</p>  
<p>Email: meomeo@gomeo.com</p>
```

- ❑ Trong view thông thường để hiển thị giá trị của biến thì dùng cú pháp `<?php echo $name; ?>`
- ❑ Trong blade view, để hiện giá trị biến/biểu thức, dùng cú pháp gọn hơn nhiều, đó là `{{ ... }}`. Ví dụ:

1. routing dẫn vào action

```
//routes/web.php
Route::get('khuyen', [ThuctapController::class, 'khuyen']);
```

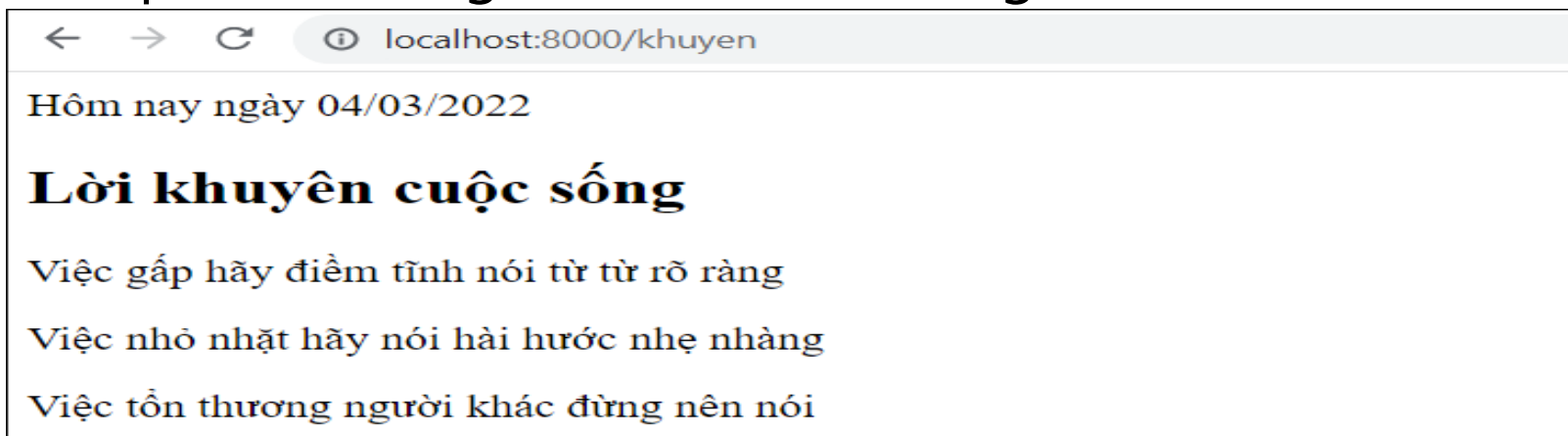
2. Trong controller, nạp view và truyền dữ liệu cho view

```
class ThuctapController extends Controller {
    function khuyen(){
        $h = gmdate('d/m/Y', time()+3600*7);
        return view('loikhuyen', ['homnay'=>$h, 'title'=>'mr']);
    }
}
```

3. Thì trong view sẽ hiện dữ liệu như sau:

```
<!-- resources/views/Loikhuyen.blade.php -->
<span>Hôm nay ngày {{$homnay}} </span>
<h2>Lời khuyên cuộc sống</h2>
<p>Việc gấp hãy điềm tĩnh nói từ từ rõ ràng</p>
<p>Việc nhỏ nhặt hãy nói hài hước nhẹ nhàng</p>
<p>Việc tổn thương người khác đừng nên nói</p>
```

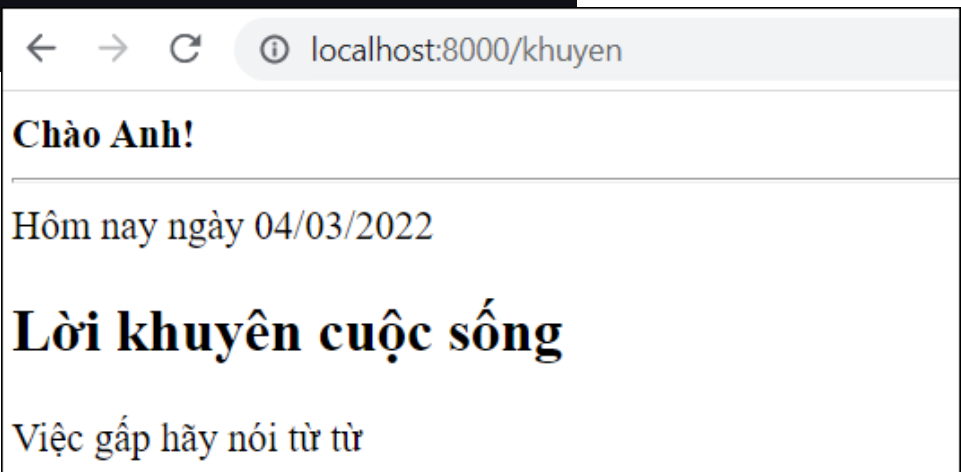
Kết quả hiển thị giá trị các biến trong view:



CÁC CÚ PHÁP HỖ TRỢ TRONG BLADE

- ❑ Blade hỗ trợ cú pháp @if để diễn tả điều kiện trong view
- ❑ @if trong blade viết rất tự nhiên, giúp hiện thông tin rất uyển chuyển

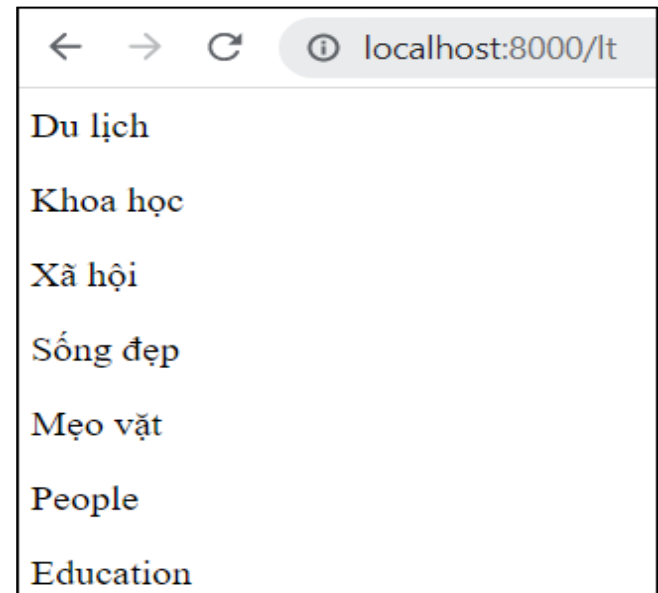
```
<!--loikhuyen.blade.php-->
@if ($title=="mr")
    <b>Chào Anh! </b>
@else
    <b>Chào Chị!</b>
@endif
<hr>
<span>Hôm nay ngày {{$homnay}}</span>
<h2>Lời khuyên cuộc sống</h2>
<p>Việc gấp hãy nói từ từ</p>
```



- ❑ Blade hỗ trợ cú pháp **@foreach ... @endforeach**, giúp lặp qua một mảng dữ liệu trong view.

```
Route::get('/lt', function() {  
    $kq = DB::table("loaitin")->select('id','ten')->get();  
    return view("loaitin", ['listloai'=>$kq]);  
});
```

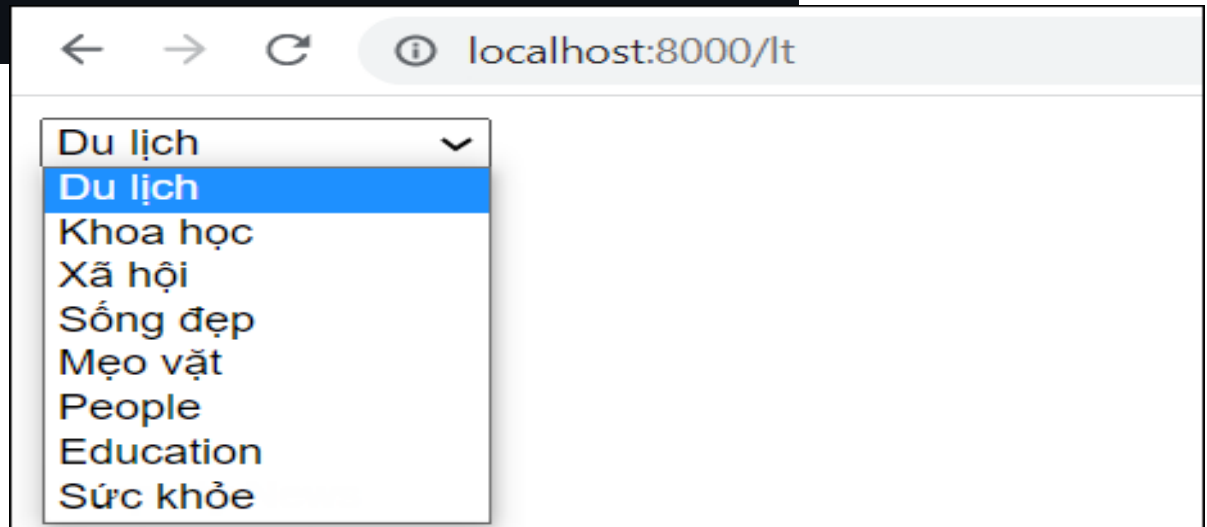
```
<!--loaitin.blade.php-->  
@foreach ($listloai as $lt)  
    <p>{{ $lt->ten }}</p>  
@endforeach  
<hr>
```



CÚ PHÁP LẶP FOR TRONG BLADE

- ❑ Blade hỗ trợ cú pháp **@for...** **@endfor** dùng để lặp trong view, giúp hiện nhiều dữ liệu rất đơn giản

```
<!--loaitin.blade.php-->
<select>
@for ($i=0; $i<count($listloai); $i++)
    <option value="{{ $listloai[$i]->id }}">
        {{ $listloai[$i]->ten }}
    </option>
@endfor
</select>
```



- ❑ Khi có nhu cầu viết code php trong blade view ,
hãy bao quanh bởi khối @php ... @endphp
- ❑ Ví dụ:

```
@php
    $soluong = 0;
    $gio = date('d/m/Y');
    echo 'Hôm nay ngày : ' . $gio;
@endphp
```

- ❑ Trong blade view, muốn nhúng view con vào thì dùng lệnh @include
- ❑ View con được chèn sẽ kế thừa tất cả dữ liệu từ view cha.
- ❑ Cũng có thể truyền vào cho view con một mảng các dữ liệu bổ sung.

```
@include('loichuc', ['str' => 'Chúc an lành'])
```



DEMO

- Demo sử dụng các cú pháp blade



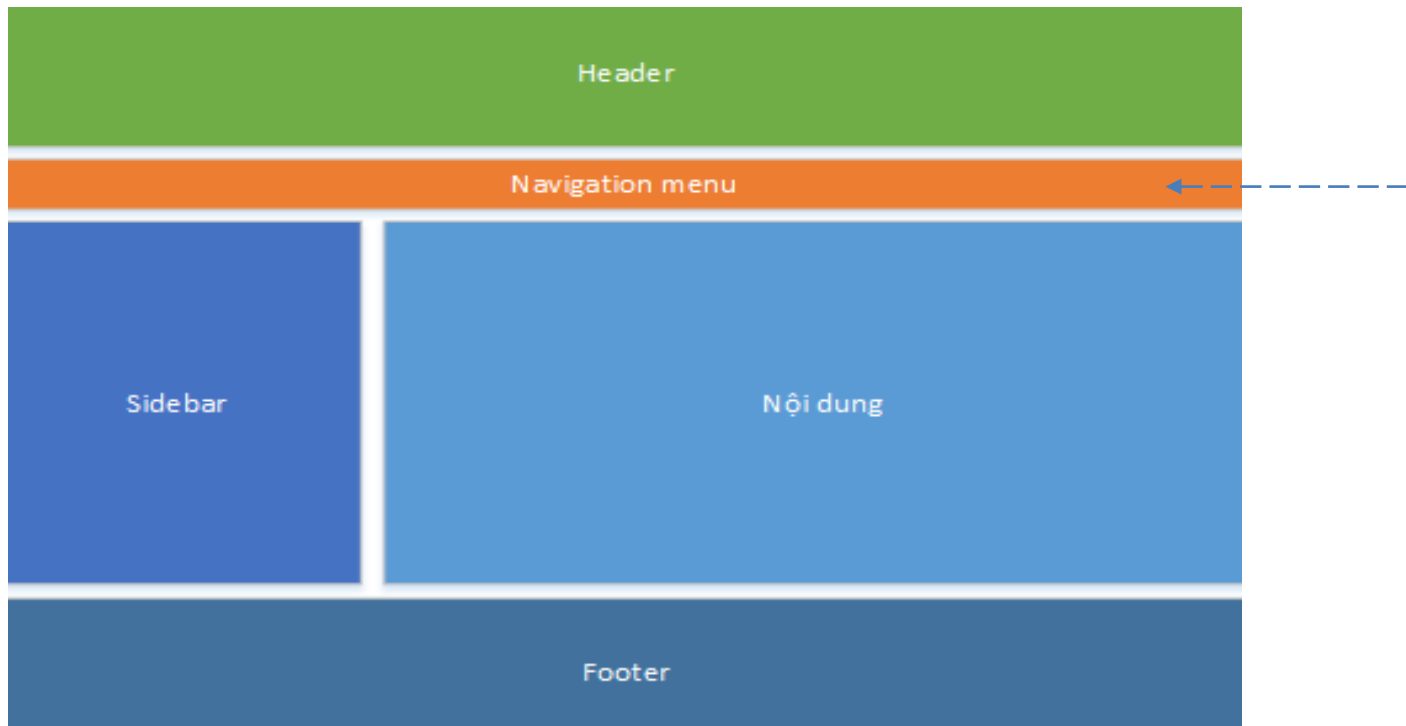


LARAVEL FRAMEWORK

BÀI 3: SỬ DỤNG BLADE TEMPLATE TRONG LARAVEL

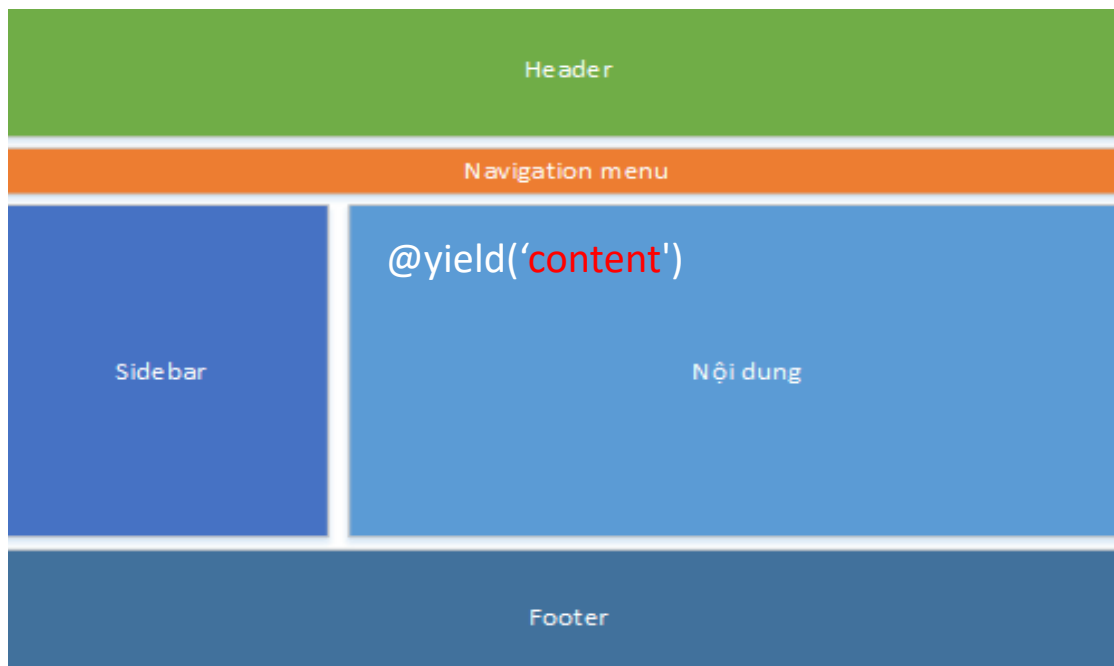
PHẦN 2

- ❑ Layout là trang khung, là mẫu dùng chung cho các trang chức năng trong website (như trang chủ, liên hệ, đăng ký, chi tiết sản phẩm...)
- ❑ Trong 1 project Laravel, có thể tạo 1 hoặc nhiều layout khác nhau



TẠO LAYOUT TRONG LARAVEL

- ❑ Trong layout, có thể định nghĩa sẵn nhiều vùng để view con đưa nội dung vào.
- ❑ Mỗi view con có thể chỉ định layout và đưa nội dung của mình vào các vùng đã định nghĩa trong layout.
- ❑ Ví dụ: lienhe có dùng layout, khi user xem, có thể thấy cả layout và nội dung lienhe được đưa nội dung vào layout.



dangky.blade.php

```
<h1>Đăng ký</h1>
```

lienhe.blade.php

```
@section('content')
<h1>Liên hệ</h1>
@section
```


- ❑ Mỗi file layout được lưu trong folder **resources/views**
- ❑ Có 2 cách tạo layout trong Laravel
 - ❖ Dùng template Inheritance – tạo layout như view lớn và các view con nhúng nội dung vào layout – là cách dùng dễ triển khai hơn
 - ❖ Tạo layout dùng component – cách này phức tạp hơn.
- ❑ Bài học sẽ hướng dẫn bạn dùng template inheritance để tạo layout
- ❑ Có nhiều lệnh đặc biệt được dùng trong trang layout và các view con để phát sinh nội dung như @extends, @yield, @section, @parent...

❑ Các lệnh đặc biệt dùng trong layout

- ❖ Lệnh **@yield('tên', 'Nội dung')**: là lệnh dùng để đặt tên cho 1 vị trí trong layout. View con dựa theo tên yield đưa nội dung vào yield trong layout.
- ❖ Lệnh **@section ... @show**: tương tự @yield, cũng là lệnh trong layout để đặt tên đánh dấu cho vị trí trong layout mà các view con sẽ đưa nội dung vào.
- ❖ Ví dụ:

```
@yield('tieudetrang', 'Bán hàng online')
```

```
@section('spNoiBat')  
    <p> Sản phẩm nổi bật</p>  
@show
```

❑ Các lệnh đặc biệt dùng trong view con

- ❖ Lệnh **@extends** : Dùng trong view con, để chỉ định layout mà view con sẽ kế thừa.
- ❖ Lệnh **@section... @endsection** : là lệnh trong view con bao quanh 1 vùng nội dung mà view con sẽ nhúng vào layout
- ❖ Lệnh **@parent** : Là lệnh trong view con, để lấy dữ liệu trong section của layout.

```
@section('spNoiBat')  
    <div class='sanpham'> SP1 </div>  
    <div class='sanpham'> SP2 </div>  
    <div class='sanpham'> SP3 </div>  
    <div class='sanpham'> SP4 </div>  
@endsection
```

❑ Ví dụ sau phối hợp các lệnh @yield , @extends, @section

1. File layout lưu trong views/app.blade.php
2. Trong đó có 2 vị trí được đặt tên là **noidung** và **sidebar**

```
<!--views/app.blade.php-->
<head><title>Shopping</title></head>
<body>
<div class="container">
  <header>HEADER</header>
  <nav><p>MENU</p></nav>
  <article>
    @yield('noidung','Nội dung chính')
  </article>
  <aside>
    @section('sidebar')
      <b>Thông tin bổ sung</b>
    @show
  </aside>
  <footer>FOOTER</footer>
</div> </body>
```

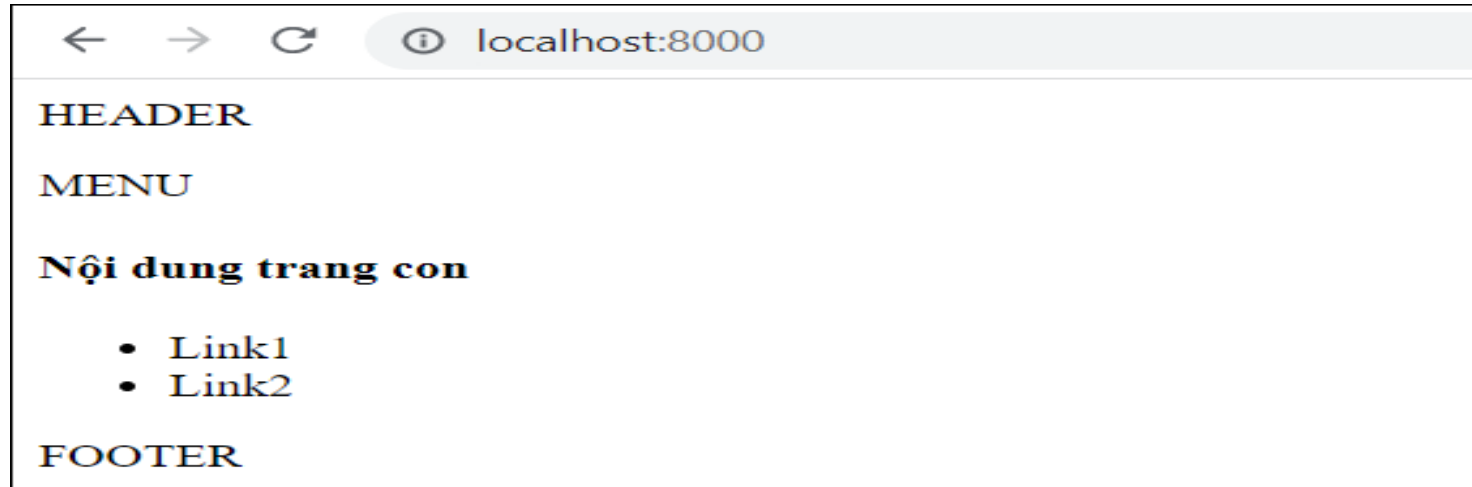
2. File view con views/child1.blade.php có sử dụng lệnh:

- ❖ Lệnh **@extends('app')** dùng để kế thừa layout **app** đã tạo.
- ❖ Lệnh **@section ... @endsection** để khai báo nội dung cho 2 vùng **noidung** và **sidebar** trong layout

```
<!--views/child1.blade.php-->
@extends('app')
@section('noidung')
    <h4>Nội dung trang con</h4>
@endsection
@section('sidebar')
    <ul>
        <li>Link1</li>
        <li>Link2</li>
    </ul>
@endsection
```

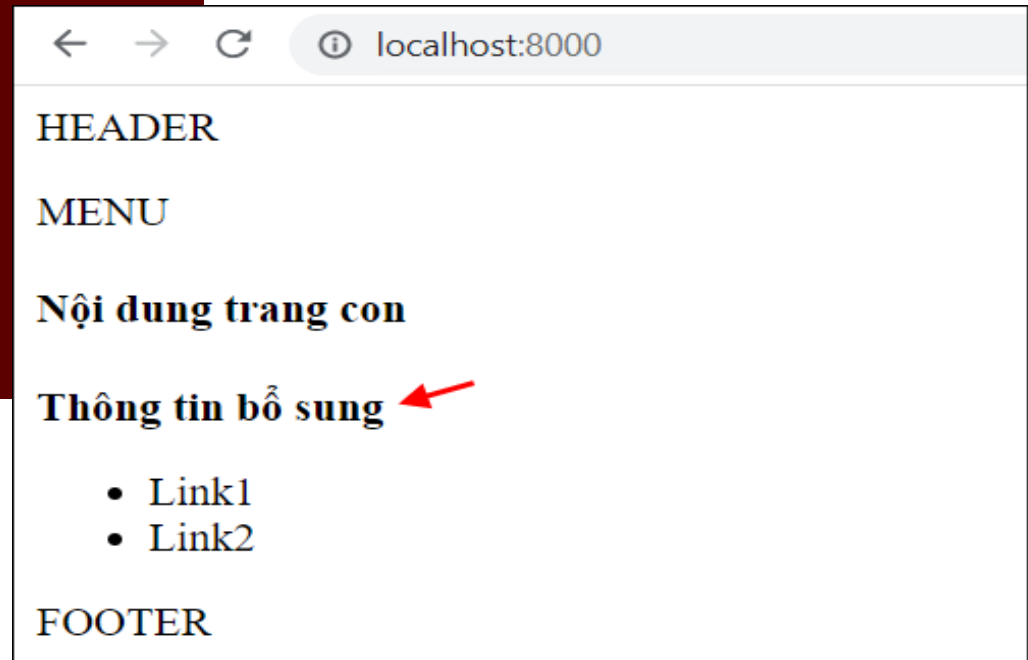
3. Routing vào view con, view con sẽ kéo layout vào và đưa nội dung vào các vùng đã định nghĩa

```
//routes/web.php
Route::get('/', function(){
    return view("child1");
});
```

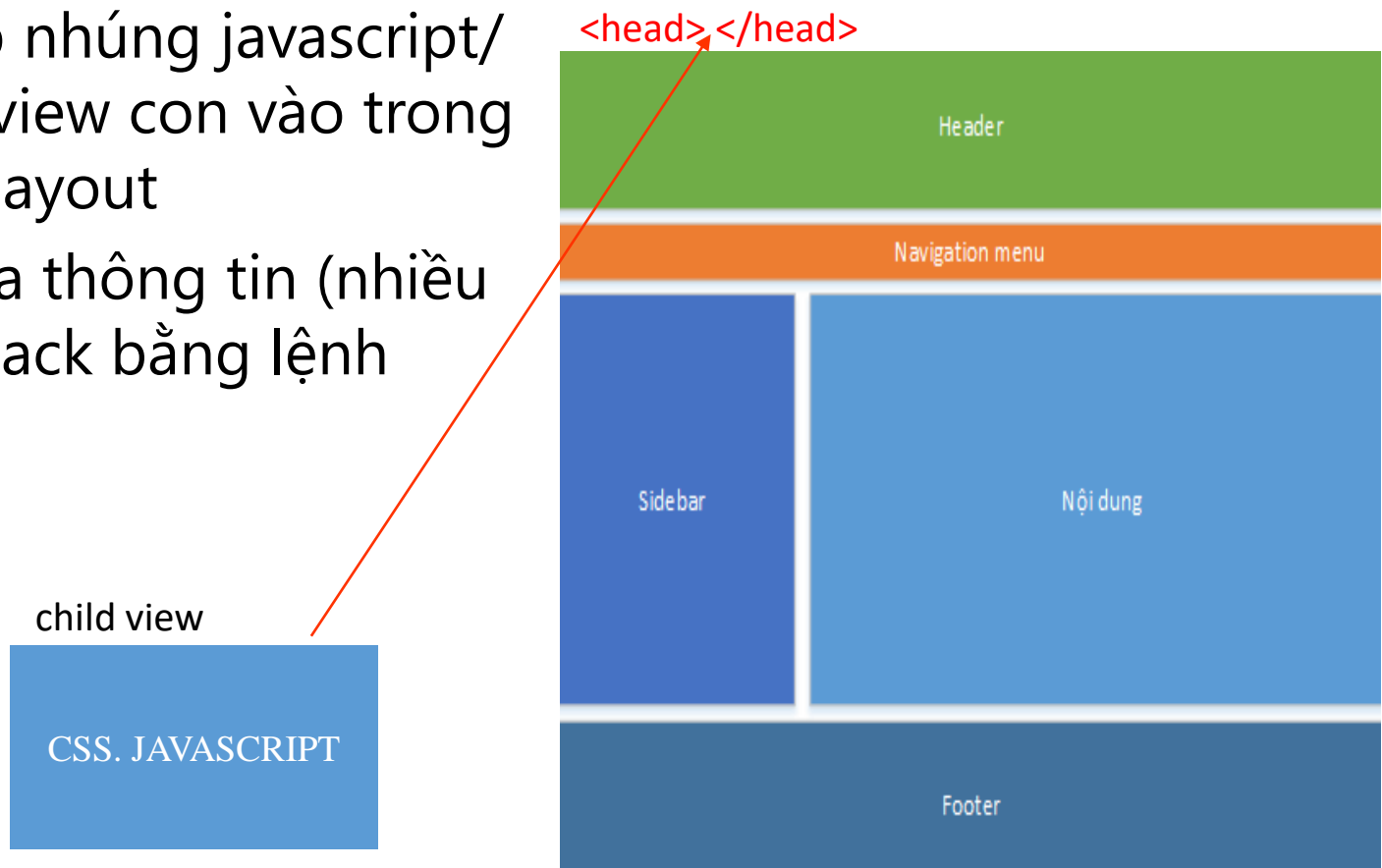


Lệnh **@parent** dùng để giữ nội dung section của layout

```
<!--views/child1.blade.php-->
@extends('app')
@section('noidung')
    <h4>Nội dung trang con </h4>
@endsection
@section('sidebar')
    @parent
    <ul>
        <li>Link1</li>
        <li>Link2</li>
    </ul>
@endsection
```



- ❑ Sử dụng stack để render thông tin vào một vị trí nào đó trong layout.
- ❑ Định nghĩa stack bằng lệnh **@stack(tênStack)**
- ❑ Stack giúp nhúng javascript/css trong view con vào trong head của layout
- ❑ Có thể đưa thông tin (nhiều lần) vào stack bằng lệnh @push



- Trong layout, định nghĩa 2 stack bằng lệnh **@stack(tênStack)**

```
<!--views/app.blade.php-->
<head>
  @stack('scripts')
  @stack('style')
</head>
```

- Đưa script, style từ view con vào stack

- Kết quả

```
<!--views/child1.blade.php-->
@push('scripts')
  <script src="/jquery.js"></script>
@endpush
@push('scripts')
  <script>alert('Chúc an lành')</script>
@endpush
@push('style')
  <style> li{ color: red} </style>
@endpush
```

→ ↻ ⓘ view-source:localhost:8000

```
<!--views/app.blade.php-->
<head>
  <script src="/jquery.js"></script>
  <script>alert('Chúc an lành')</script>
  <style> li{ color: red} </style>
</head>

<body>
<div class="container">
  <header>HEADER</header>
  <nav><p>MENU</p></nav>
```

- ❑ Form là thành phần không thể thiếu khi thiết kế view cho website. Form giúp nhận thông tin từ người dùng.
- ❑ **CSRF Field:** Là field ẩn (hidden) mà Laravel bắt buộc nhúng vào form cho mục đích bảo mật, giúp bảo vệ website.
- ❑ Sử dụng lệnh **@csrf** ở bất cứ chỗ nào trong form để nhúng field csrf vào form.

```
<form action="" method="POST">  
  @method('PUT')  
  ...  
</form>
```

- ❑ **Method Field:** là field ẩn mà bạn chèn thêm vào form khi có nhu cầu sử dụng các method http như PUT, PATCH, DELETE.

```
<form method="POST" action="">  
  @csrf  
  ...  
</form>
```










DEMO






- Tạo và sử dụng master page
- Xây dựng Blade dùng Control structure



Phần I:

-  Giới thiệu về Blade
-  Hiện dữ liệu trong Blade
-  Các cú pháp hỗ trợ trong Blade
-  Lặp foreach qua 1 mảng trong Blade
-  Cú pháp lặp for trong Blade
-  Viết mã php trong blade
-  Nhúng view con và blade view

Phần II:

-  Tạo layout trong Laravel
-  Tạo layout với template Inheritance
-  Sử dụng các lệnh trong layout
-  Sử dụng stack
-  Sử dụng form



- ☐ <https://laravel.com/docs/views>
- ☐ <https://laravel.com/docs/blade>
- ☐ <https://laravel.com/docs/blade#forms>



Cảm ơn