```
</ul>
```

Route postيتم استخدامه لاستقبال البيانات من المستخدم المدخلة عن طريق form

لذلك نحتاج إضافة form في الصفحة

```
<form action="/users" method="POST">
@csrf
<input type="text" name="user_name" id="">
<input type="submit" name="user_sub" id="">
</form>
```

ونكتب ال Route بهذه الطريقة

```
Route::post("users",function(Request $request)
    {
        return $request;
    }
);
```

تمرير بيانات باستخدامget

```
//Route::get("users/{id}",function(int $id)or ($id)
//نحدد النوع نجبر المستخدم على نوع بيانات واحد
Route::get("users/{name}",function(string $name)
    {
        if($name==="Ahmed")
        return "admin";
        else {
            return "user";
        }
    }
);
```

- **Blade Template**

Bladeهو محرك القوالب البسيط والقوي المضمن في Laravel على عكس بعض محركات قوالب PHP، لا يمنعك Blade من استخدام كود PHP العادي في القوالب الخاصة بك، في الواقع يتم تجميع جميع قوالب Blade في كود PHP عادي وتخزينها مؤقتا حتى يتم تعديلها.

Blade يقوم باختصار الأكواد بدل من كتابتها كما فيphp

نضيفrouter users

```
Route::get("users",function()
    {
        $username="Ahmed";
                return view('users',compact('username'));
    });
//or
Route::get('users ', function () {
    return view('welcome', [' username ' => ' Ahmed ']);
});
```

انشاء صفحةview users.blade.php

```
{{$username}}
@if ($username=="Ahmed")
<br>welcome Ahmed
@else
hello
@endif
@for ($i=0;$i<10;$i++)
{{$i}}
@endfor
```

هنا blade يقوم بتحويل هذا الكود الى شفرة php نستطيع رؤيته من مجلد files<-views<-storage

```
{{-- بدلا من كتابتها في وسم php --}}

<?php
echo $username;
if($username=="Ahmed")
echo "<br>welcome Ahmed";
else {
    echo "welcome";
}
```

يتم إرسال عبارات {{}} Blade تلقائيًا من خلال وظيفة htmlspecialchars في PHP لمنع هجمات XSS.

```
The current timestamp is {{ time() }}.
```

بشكل افتراضي، يتم إرسال عبارات {{}} Blade تلقائيًا من خلال وظيفة htmlspecialchars في PHP لمنع هجمات XSS. إذا كنت لا تريد هروب بياناتك ، فيمكنك استخدام الصيغة التالية:

```
{!! $username !!}
```

يمكنك إنشاء عبارات if باستخدام توجيهات if و elseif و else و endif. تعمل هذه التوجيهات بشكل مماثل لمثيلاتها في PHP

```
@if (count($records) === 1)
I have one record!
@elseif (count($records) > 1)
I have multiple records!
@else
I don't have any records!
@endif
```

يمكن استخدام التوجيهات isset و empty كاختصارات ملائمة لوظائف PHP الخاصة بهم

```
@isset($records)
    // $records is defined and is not null...
@endisset

@empty($records)
    // $records is "empty"...
@endempty
```

توجيهات المصادقة Authentication Directives

يمكن استخدام التوجيهين auth و guest لتحديد ما إذا كان المستخدم الحالي مصادقًا أم ضيفًا:

```
@auth
    // The user is authenticated...
@endauth

@guest
    // The user is not authenticated...
@endguest
```

يمكن إنشاء عبارات التبديل باستخدام توجيهات switch و case و break و default و endswitch:

```
@switch($i)
    @case(1)
        First case...
        @break
```

```
    @case(2)
        Second case...
        @break

    @default
        Default case...
@endswitch
```

LOOPS

بالإضافة إلى العبارات الشرطية ، يوفر Blade توجيهات بسيطة للعمل مع هياكل حلقة PHP. مرة أخرى ، تعمل كل من هذه التوجيهات بشكل مماثل لنظيراتها في PHP:

```
@foreach ($users as $user)
    <p>This is user {{ $user->id }}</p>
@endforeach
//============================
@forelse ($users as $user)
    <li>{{ $user->name }}</li>
@empty
    <p>No users</p>
@endforelse
/==================================
@while (true)
    <p>I'm looping forever.</p>
@endwhile
```

أثناء التكرار خلال حلقة foreach ، يمكنك استخدام متغير الحلقة للحصول على معلومات قيمة حول الحلقة ، مثل ما إذا كنت في التكرار الأول أو الأخير من خلال الحلقة.

عند استخدام الحلقات ، يمكنك أيضًا تخطي التكرار الحالي أو إنهاء الحلقة باستخدام التوجيهين continue و break:

```
@foreach ($users as $user)
    @if ($user->type == 1)
        @continue
    @endif

    <li>{{ $user->name }}</li>
```

```
    @if ($user->number == 5)
        @break
    @endif
@endforeach
```

يمكنك أيضًا تضمين شرط استمرار أو كسر في بيان التوجيه:

```
@foreach ($users as $user)
    @continue($user->type == 1)

    <li>{{ $user->name }}</li>

    @break($user->number == 5)
@endforeach
```

The Loop Variable

أثناء التكرار خلال حلقة foreach ، سيكون متغير $ loop متاحًا داخل الحلقة. يوفر هذا المتغير إمكانية الوصول إلى بعض أجزاء المعلومات المفيدة مثل فهرس الحلقة الحالية وما إذا كان هذا هو التكرار الأول أو الأخير من خلال الحلقة:

```
@foreach ($users as $user)
    @if ($loop->first)
        This is the first iteration.
    @endif

    @if ($loop->last)
        This is the last iteration.
    @endif

    <p>This is user {{ $user->id }}</p>
@endforeach
```

إذا كنت في حلقة متداخلة ، يمكنك الوصول إلى متغير حلقة الأصل $loop عبر الخاصية الأصل

```
@foreach ($users as $user)
    @foreach ($user->posts as $post)
        @if ($loop->parent->first)
            This is the first iteration of the parent loop.
        @endif
    @endforeach
@endforeach
```

يحتوي المتغير $ loop أيضًا على مجموعة متنوعة من الخصائص المفيدة الأخرى:

وصف الخصائص

| | |
|---|---|
| $loop->index | فهرس تكرار الحلقة الحالية (يبدأ من 0). |
| $loop->iteration | تكرار الحلقة الحالية (يبدأ عند 1). |
| $loop->remaining | التكرارات المتبقية في الحلقة |
| $loop->count | إجمالي عدد العناصر في المصفوفة التي يتم تكرارها |
| $loop->first | ما إذا كان هذا هو التكرار الأول للحلقة |
| $loop->last | ما إذا كان هذا هو آخر تكرار خلال الحلقة |
| $loop->even | ما إذا كان هذا تكرارًا زوجيا خلال الحلقة |
| $loop->odd | ما إذا كان هذا تكرارًا فرديًا خلال الحلقة |
| $loop->depth | مستوى تداخل الحلقة الحالية |
| $loop->parent | عندما تكون في حلقة متداخلة ، متغير حلقة الأصل او الاب |

- **التخطيطات باستخدام وراثة النموذج: layouts using Template Inheritance**

يمكن إنشاء التخطيطات عبر "وراثة النموذج" ، عندما يكون هناك أكواد مكرره نريد استخدامها في أكثر من صفحة نقوم بإنشاء مجلد Layouts في views ونكتب الكود الذي نحتاجه.

للبدء ، دعنا نلقي نظرة على مثال بسيط. أولاً، سوف نفحص تخطيط الصفحة. نظرًا لأن معظم تطبيقات الويب تحافظ على نفس التخطيط العام عبر صفحات مختلفة، فمن الملائم تحديد هذا التخطيط كعرض Bladeفردي:

```
<!-- resources/views/layouts/app.blade.php -->
<html>
    <head>
        <title>App Name - @yield('title')</title>
    </head>
    <body>
        @section('sidebar')
            This is the master sidebar.
```

```
        @show
        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

للتوضيح اكثر تطبيق عملي لاكثر من صفحة:

نضيف عدة صفحاتwelcome.balde,users.blade,posts.blade

في Routes بملف web.php نكتب مايلي:

```
Route::get('/', function () {
    return view('welcome');
});
Route::get('users', function () {
    return view('users');
});
Route::get('posts', function () {
    return view('posts');
});
```

نستخدم أسماء Routes في الروابط للتنقل بين الصفحات

نضيف مجلد باسم layouts داخل مجلد views ومن ثم يتم إضافة صفحة ولتكن باسم header.blade.phpونضيف محتوى لهذه الصفحة

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
{{-- المحتوى من خلال هذا المتغير يتم تغييره في الصفحات --}}
{{-- @yield('content', 'Default content') --}}
<title>@yield('title','name web')</title>
</head>
<body>
<header>
<nav>
    <a href="/">main</a>
    <a href="/users">users</a>
```

```
    <a href="/posts">posts</a>
</nav>
</header>
<aside>
   aside
   {{-- عرضه في جميع الصفحات وممكن تعديله او اضافه اليه --}}
   @section('sidebar')
show sidebar
   @show
</aside>
<main>
@yield('main_content')
</main>
<footer>
footer content
</footer>
</body>
</html>
```

يتم استخدام الامر التالي في الصفحات الأخرى لتغيير محتوى جزء معين

```
@yield('content', 'Default content')
```

عرض محتوى ويمكن تغييره في كل صفحة نكتب الامر التالي

```
@section('sidebar')
show sidebar
@show
```

في الصفحات الأخرى نقوم باستدعاء هذا الملف باستخدام

@extends('layouts.header')

```
{{-- welcome.blade.php --}}
@extends('layouts.header')
{{-- يأخذ القيمة الافتراضية ل yield --}}
{{-- @section('title')
الرئيسية
@endsection --}}
@section('main_content')
مقالات مثلا
@endsection
{{-- تبديل محتواه عن الموجود في ملف header او اضافة له عن طريق parent --}}
@section('sidebar')
```

```
@parent
changes
@endsection
{{-- users.blade.php --}}
{{-- تضمين ملف وراثة وممكن احدد ماذا يعرض في مكان معين تغيير المحتوى --}}
@extends('layouts.header')
@section('title')
users
@endsection
@section('main_content')
نموذج انشاء حساب
@endsection
{{-- posts.blade.php --}}
@extends('layouts.header')
@section('title')
posts
@endsection
@section('main_content')
main posts
@endsection
10-
```

- **وحدات التحكمControllers**

انشاء وحدة تحكمController

نستخدم terminal نكتب الامر

```
php artisan make:controller NameController
```

حيث NameController اسم controller المراد انشاءه ويفضل ان يتم كتابته بهذه الطريقة أول حرف capital من الكلمة الأولى وأيضا اول حرف capital لكلمةcontroller

يتم إضافته في المسار App/Http/Controllers/

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class PostController extends Controller
{
public function showUsers()
```