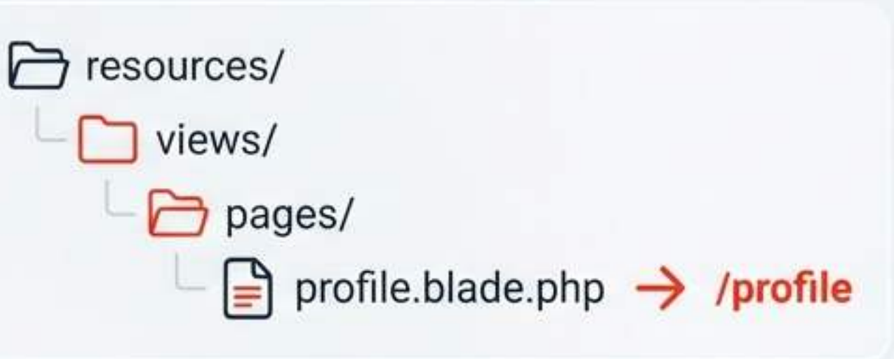
Laravel Folio

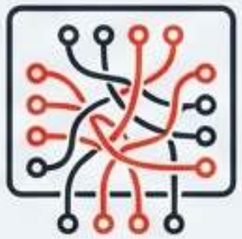
تحويل مسارات التطبيق من الأكواد إلى الصفحات

عالمان من تعريف المسارات (Routing)

التوجيه المعتمد على الصفحات (Page-Based Routing)

يتم إنشاء المسار تلقائيًا بمجرد إنشاء ملف Blade في مجلد معين.





التوجيه المعتمد على الأكواد (Code-Based Routing)

يتم تعريف كل مسار بشكل صريح في ملف `routes/web.php`.

```php
// routes/web.php
use App\Http\Controllers\ProfileController;

Route::get('/profile', [ProfileController::class, 'show']);
```

التثبيت والبدء في دقيقتين

1 الخطوة ١: التثبيت عبر Composer

```
composer require laravel/folio
```

هذا الأمر يقوم بتنزيل الحزمة وإضافتها لمشروعك.

2 الخطوة ٢: تشغيل أمر التثبيت

```
php artisan folio:install
```

هذا الأمر يقوم بإنشاء `FolioServiceProvider` وتهيئة مجلد `resources/views/pages` الذي سيبحث فيه Folio عن صفحاتك.

بعد هاتين الخطوتين، أنت جاهز تمامًا لبدء بناء صفحاتك.

صفحتك الأولى: المسارات الثابتة (Static Routes)

١. أنشئ الملف

resources/views/pages/
about.blade.php

٢. النتيجة: مسار فوري

٣. اكتب الكود

```
<x-layout>
    <h1>About Us Page</h1>
</x-layout>
```

معلومة هامة: Folio يعمل كإضافة لنظام الـ routing الحالي. المسارات في `web.php` لا تزال تعمل بشكل طبيعي.

تنظيم مساراتك: التداخل والفهرسة (Nesting & Indexing)

صفحات الفهرس (Index Routes)

الفكرة: ملف `index.blade.php` يصبح المسار الرئيسي للمجلد الموجود فيه.

المسارات المتداخلة (Nested Routes)

الفكرة: أنشئ مجلدات لإنشاء مسارات متداخلة.

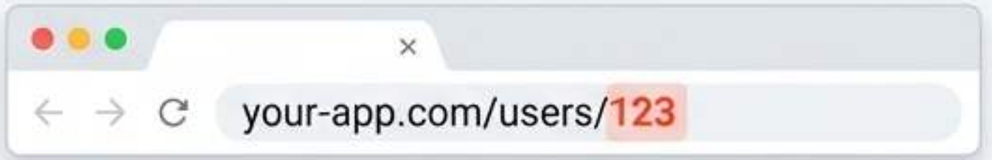
أداة مساعدة: لمعرفة كل مسارات Folio في مشروعك، استخدم الأمر:
`php artisan folio:list`

نحو الديناميكية: متغيرات المسار (Route Parameters)

الفكرة: استخدم الأقواس المربعة `[]` في اسم الملف لالتقاط جزء من الرابط.

pages/
users/
[id].blade.php

الملف: pages/users/[id].blade.php



مثال الرابط: your-app.com/users/123

الوصول للمتغير في الكود: المتغير **$id** يصبح متاحًا تلقائيًا داخل ملف الـ Blade.

```
<!-- The $id variable is automatically available -->
<div>
    <h1>Showing profile for User ID: {{ $id }}</h1>
</div>
```

المتغير $id

ميزة إضافية لالتقاط أجزاء متعددة (مثل /tags/php/)، استخدم [...slugs].blade. سيتم تمريرها كـ array.

السحر الحقيقي: الربط التلقائي للـ Models (Route Model Binding)

الطريقة التقليدية (داخل الصفحة)

```
@php
    $user = App\Models\User::findOrFail($id);
@endphp

<h1>{{ $user->name }}</h1>
```

طريقة Folio السحرية

السر: قم بتسمية الملف باسم الـ Model (بحرف كبير).

الملف: `pages/users/[User].blade.php`

الرابط: `your-app.com/users/1`

الكود: الـ `Model` يتم حقنه تلقائيًا في الصفحة وجاهز للاستخدام.

```
<!-- No PHP needed! The $user object is
injected automatically -->
<h1>Hello, {{ $user->name }}</h1>
```

تخصيص ربط الـ Models المتقدم

١. استخدام مفتاح مخصص (Custom Key)

- **المشكلة:** 'أريد جلب المستخدم عن طريق `username` وليس `id`".
- **الحل:** حدد العمود في اسم الملف بعد اسم الـ Model.

الملف:
`pages/users/[User:username].blade.php`

`/users/elon-mask`

ملاحظة: في نظام Windows، استخدم `-` بدلاً من `:`، مثال: `[User-username].blade.php`.

٢. تحديد مسار الـ Model (Model Location)

- **المشكلة:** "الـ Models الخاصة بي ليست في مجلد `app/Models` الافتراضي."
- **الحل:** قم بتوفير الـ namespace الكامل للـ Model.

الملف:
`pages/users/[.App.Domain.Users.Models.User].blade.php`

تأمين صفحاتك: تطبيق الـ Middleware

الطريقة الأولى: داخل ملف الصفحة (For a single page)

استخدم دالة `middleware()` في بداية ملف الـ Blade.
هذه الطريقة مثالية لصفحة واحدة تحتاج حماية خاصة.

```php
<?php
use function Laravel\Folio\{middleware};
middleware(['auth', 'verified']);
?>

<div>Your Dashboard</div>
```

الطريقة الثانية: لمجموعة صفحات (For a group of pages)

قم بتعريف الـ Middleware في ملف `FolioServiceProvider`
لتطبيقه على مجموعة من المسارات التي تتبع نمطًا معينًا.

```php
// In FolioServiceProvider.php
Folio::path(resource_path('views/pages/admin'))
    ->uri('/admin')
    ->middleware([
        '*' => ['auth', 'verified'],
    ]);
```

تسهيل توليد الروابط: الأسماء الموجهة (Named Routes)

١. تسمية المسار (Naming the route)

استخدم دالة `name()` في بداية ملف الـ Blade.

```
// In pages/users/index.blade.php
<?php
use function Laravel\Folio\{name};
name('users.index');
?>
```

٢. استخدام الاسم (Using the name)

استخدم دالة `route()` المعتادة في أي مكان في تطبيقك.

```
<a href="{{ route('users.index') }}">All Users</a>
```

مع تمرير المتغيرات

```
// For a route like [User].blade.php named 'users.show'
route('users.show', ['user' => $user]);
```

تحكم كامل: استخدام خطافات العرض (Render Hooks)

الفكرة: دالة `render` تتيح لك اعتراض عملية العرض لتنفيذ منطق مخصص.

حالة الاستخدام: التحقق من صلاحيات المستخدم قبل عرض تفاصيل منشور.

```php
<?php
use Illuminate\View\View;
use App\Models\Post;
use function Laravel\Folio\render;

render(function (View $view, Post $post) {
    // Check if the user is authorized to view the post
    if (! Auth::user()->can('view', $post)) { 1
        return response('Unauthorized', 403);
    }

    // Or add extra data to the view
    return $view->with('photos', $post->author->photos); 2
});
?>

<h1>{{ $post->title }}</h1>
```

1 تنفيذ منطق تحكم (مثل التحقق من الصلاحيات)

2 إضافة بيانات إضافية للـ View

الخلاصة: احصل على قوة الـ Controller مباشرة في صفحتك عندما تحتاج إليها.

إعدادات متقدمة للتطبيقات المعقدة

١. مسارات متعددة (Multiple Page Directories)

يمكنك تحديد مجلدات صفحات مختلفة مع مسارات URI مختلفة. مختلفة. مثالي لفصل لوحة التحكم عن واجهة الموقع.


```
// In FolioServiceProvider
Folio::path(resource_path('views/pages/guest'))->uri('/');
Folio::path(resource_path('views/pages/admin'))->uri('/admin');
```

٢. توجيه النطاقات الفرعية (Subdomain Routing)

يمكنك توجيه نطاق فرعي معين (example.com.{admin}) إلى مجلد صفحات مخصص.


```
// In FolioServiceProvider
Folio::domain('{account}.example.com')
        ->path(resource_path('views/pages/account'));
```

متى يجب أن تستخدم Folio؟

مثالي لـ (Ideal for)

صفحات التسويق والمحتوى: مثل صفحة "من نحن"، "الأسعار"، "سياسة الخصوصية".

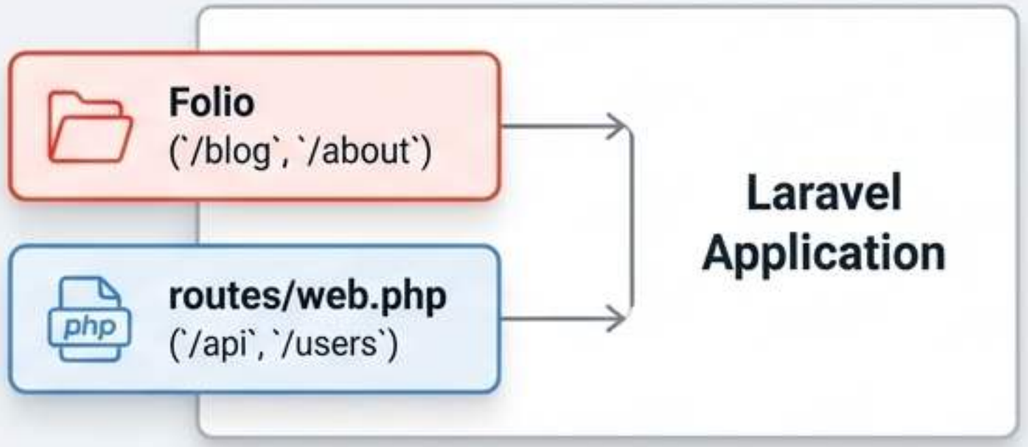
المدونات (Blogs): بنية الملفات تتناسب تمامًا مع بنية المدونة (قائمة مقالات، عرض مقال).

لوحات التحكم (Admin Panels): حيث يكون كل قسم في لوحة التحكم صفحة منفصلة.

النماذج الأولية السريعة (Rapid Prototyping).

نقطة قوة رئيسية: الدمج (The Hybrid Approach)

لست مضطرًا للاختيار. يمكنك استخدام Folio لجزء من تطبيقك (مثل (`/blog`) بينما تظل بقية الأجزاء (مثل منطق الـ API المعقد) تستخدم `routes/web.php` التقليدي.



"استخدم الأداة المناسبة للمهمة المناسبة."

أفضل الممارسات لتحقيق أقصى استفادة

٣. فكر في Folio أولاً (Folio-First Mindset)

لماذا؟: للمسارات التي تعرض واجهة مستخدم، ابدأ بالتفكير 'هل يمكن عمل هذا بملف Folio؟". هذا يقلل من التنقل بين ملفات الـ routes والـ controllers والـ views.

٢. التنظيم هو مفتاح النجاح (Organization is Key)

لماذا؟: لأن بنية ملفاتك هي نفسها بنية مساراتك. حافظ على مجلد `pages` منظمًا وواضحًا.

١. استخدم دائمًا تخزين الـ Routes المؤقت (Always Use Route Caching)

لماذا؟: لحصول على أقصى أداء في بيئة الإنتاج. Folio مصمم ليعمل بكفاءة مع نظام التخزين المؤقت في Laravel.

```
php artisan route:cache
```

Folio: حيث يلتقي الكود بالوضوح

يقدم Laravel Folio طريقة أكثر سهولة وتنظيمًا للتعامل مع جزء كبير من مسارات تطبيقك. من خلال تقريب الـ view من الـ route، فإنه يقلل من التعقيد ويسمح لك بالتركيز على بناء ميزات رائعة بشكل أسرع. إنه ليس مجرد توجيه قائم على الملفات؛ إنه إعادة تفكير في كيفية بناء تطبيقات الويب بشكل بديهي.