إنشاء نظام مصادقة المستخدمين

Authentication

في معظم تطبيقات الويب ستحتاج حتماً لإنشاء نظام استيثاق أو ما يسمى بنظام المصادقة Authentication للتثبت من هوية المستخدمين .

و لإنشاء نظام المصادقة ستعمل على عدة أمور مثل إنشاء جداول لحفظ بيانات المستخدمين ، ثم إنشاء واجهات للتسجيل و الدخول للنظام ، و بعدها كتابة التعليمات البرمجية الخاصة بالتحقق من هوية المستخدم و استعادة كلمة المرور و غير ذلك .

من أفضل المزايا في Laravel Framework احتوائها على نظام مصادقة كامل يشمل جميع الأمور السابقة ، و يتم تفعيله عن طريق الـartisan .

ملف التكوين الخاص بنظام المصادقة موجود بداخل المجلد Config بالاسم Auth.php

بالنظر في هذا الملف ستجد أن لارافل يستخدم النموذج (Model) المسمى User للمصادقة بشكل افتراضي ، و سيتعامل مع هذا الجدول حسب نظام Eloquent

```
    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\User::class,
        ],
```

يمكنك تغيير هذه الافتراضيات في حال أردت ذلك .

و في حال كان تطبيقك يشمل أكثر من جدول للاستيثاق ستحتاج لإضافة مصدر آخر هنا

على سبيل المثال إذا كنت ستنشئ جدول منفصل لـadmins فعليك تعريف الإعدادات الخاصة به في هذا الملف auth.php ، ستحتاج لإضافة provider جديد بهذا الشكل

```
        'admins' => [
            'driver' => 'eloquent',
            'model' => App\Admin::class,
        ],
```

و كذلك ستقوم بتعريف المصدر الجديد ضمن المصفوفة guards كالتالي

```
        'admin' => [
            'driver' => 'session',
            'provider' => 'admins',
        ],
```

هذه هي الفكرة العامة .. و غالباً ستكتفي بمصدر واحد و لن تحتاج إلى إجراء تغييرات عليه .

بالنظر بداخل الدليل Controllers ستجد مجلد بالإسم Auth . يتضمن هذا المجلد عددًا من المتحكمات الخاصة بنظام الاستيثاق

ForgotPasswordController لمعالجة رسائل البريد الالكتروني الخاصة بإعادة تعيين كلمة المرور

LoginController يتعلق بمصادقة المستخدم و فيه يتم تحديد الصفحة التي سيتم التوجه إليها بعد تسجيل الدخول مباشرةً

RegisterController يقوم بمعالجة عملية تسجيل المستخدمين

ResetPasswordController مسؤول عن معالجة طلبات إعادة تعيين كلمة المرور

سنقوم الآن بإنشاء جدول للمستخدمين في قاعدة البيانات ، و كما ترى فملفات التهجير الخاصة بالاستيثاق موجودة على المجلد database ، قبل إنشاء هذا الجدول (users) عليك التأكد من أنه يشمل الحقل remember_token بطول ١٠٠ حرف ، يتم تحديث هذا الحقل برمز جديد بعد كل عملية تسجيل دخول ما يزيد من أمان جلسة المستخدم .

عن طريق موجه الأوامر سننفذ أمر التهجير لإنشاء الجداول في قاعدة البيانات

```
$ php artisan migrate
```

سنقوم الآن بتشغيل الأمر الخاص بإنشاء كل ما يتعلق بعملية الاستيثاق Authentication من ملفات العروض views و المسارات Routes الخاصة بها .

عبر موجه الأوامر سننفذ الأمر التالي :

```
$ php artisan make:auth
```

لاحظ الآن ظهور ملفات جديدة ضمن الدليل views ، و بمجرد النظر ستعرف أنها الواجهات الخاصة بإنشاء المستخدمين و تسجيل الدخول للنظام

الـview المسمى home سيتم التوجه إليه مباشرةً عند الدخول للنظام .
جميع ملفات العرض الموجودة ضمن المجلد **auth** تأخذ نفس القالب المسمى app و الموجود ضمن المجلد layout ، و يمكنك تغيير هذا التصميم حسب ما تريده .

في الملف web.php الخاص بالمسارات ستلاحظ ظهور الطريقة Auth::routes() التي تعمل على توليد كافة المسارات المطلوبة ، يمكنك مشاهدة هذه المسارات بتشغيل الأمر `php artisan route:list` على موجه الأوامر .

الآن جميع الأمور جاهزة
قم بطلب العنوان register للبدء بإنشاء مستخدم جديد و لاحظ أنه لا يقبل أي كلمة مرور بطول أقل من ٦ خانات ، يمكنك تغيير هذا الإعداد من خلال الملف RegisterController .. بعد الإنتهاء من إنشاء الحساب سيتم التوجه مباشرة للصفحة Home .

يمكنك تغيير الوجهة الافتراضية من خلال الملف LoginController إما بتحديد عنوان المسار في الخاصية $redirectTo

```
protected $redirectTo = '/index';
```

أو بإعادة كتابة الدالة redirectTo() بالشكل التالي

```
protected function redirectTo()
{
    return '/index';
}
```

علمًا أن استخدامك لهذه الدالة سيلغي ما هو معرَف ضمن الخاصية redirectTo حيث أن الأولوية ستكون للدالة redirectTo().

و لاستخدام اسم المستخدم بدلاً عن البريد الإلكتروني في عملية المصادقة فقط قم بإضافة الدالة username لتعيد اسم العمود الذي تريد التحقق بواسطته

```
    public function username()
    {
        return 'name';
    }
```

أيضًا عليك التأكد من اسم الحقل في ملف العرض .. عليك تغييره إلى name بنفس اسم العمود في قاعدة البيانات ، و بالطبع **لن** يكون نوع الحقل email .

لاحظ الآن أنه سيرفض الدخول بعنوان البريد الإلكتروني ، و يتم قبول اسم المستخدم .

أحياناً قد تحتاج للوصول إلى بيانات المستخدم الذي سجل دخوله ، يمكنك استخدام الواجهة Auth للوصول لهذه البيانات بالطريقة :

```
    return Auth::user()->name ;
```

طريقة أخرى للوصول لنفس البيانات السابقة عبر الكائن Request بالشكل التالي :

```
    public function update(Request $request)
    {
        return $request->user() ;
    }
```

حماية المسارات Routes :

حالياً سيتمكن المستخدم من الوصول لأي صفحة ضمن التطبيق ..كما تلاحظ .. فكيف تقوم بحماية هذه الصفحات بحيث لا يتمكن من عرضها إلا المستخدمين المصادق عليهم فقط ؟

الحل هو استخدام Middleware لحماية الصفحات ، سيتضمن هذا الـMiddleware التعليمات البرمجية للتحقق من كون المستخدم الحالي تمت المصادقة عليه أم لا ثم تمرير الطلب أو رفضه .

هناك Middleware جاهز للتحقق من هذه الأمور ، و هو مسجل في ملف Kernal.php بالاسم Auth كما ترى

```
protected $routeMiddleware = [
'auth' => \Illuminate\Auth\Middleware\Authenticate::class,
```

```
'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
'can' => \Illuminate\Auth\Middleware\Authorize::class,
'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
];
```

و الآن لتطبيق هذا الـMiddleware على صفحات معينة ، عليك فقط إرفاقها ضمن المسار الخاص بالصفحة المطلوبة كما تعلمت سابقاً .

```
Route::resource('posts','PostController')->middleware('auth');;
```

هناك طريقة أخرى و تكون بتضمينها في ملف الـController بهذا الشكل :

```
    public function __construct()
    {
        $this->middleware('auth');
    }
```

إذا حاولت الآن استدعاء أي Function ضمن هذا المتحكم ، سيتم منعك و إعادة التوجيه إلى الصفحة الخاصة بتسجيل الدخول .