يعتبر رفع الملفات والصور واحدة من أهم الإمور التي نتعامل معها في جميع التطبيقات، ويجب الإهتمام بها لأنها تؤثر على حماية الموقع، كما أن سوء إستخدام رفع الصور والملفات سوف يؤثر على سرعة وأداء الموقع، في هذه المقالة محاولة لإلقاء نظرة على كيفية رفع الملفات بالطريقة المبسطة وكيفية التحكم في الإمتدادات ومعالجة الصور أثناء الرفع من أجل حماية وأداء أفضل.
تخيل أن لدينا الفورم التالي لإضافة مؤلف جديد (الإسم الأول first_name، الإسم الثاني second_name، الصوره avatar)
يحتوي على صوره يتم رفعها Requestلإدخال البيانات، بداية ننشئ المؤلف ومن ثم إذا كان الـ وتعديل الحقل الذي تم إضافته من أجل تسجيل إسم الصوره مع إسم المجلد الذي تم تخزين الصوره بداخلها.

ماذا لو كنا لا نريد أن يكون إسم الملف/ الصوره طويل ، بل نريد أن يكون user_id الذي تم إنشاؤه.

- يجب فصل file request في متغير حتى نستطيع الوصول للـ objects.
- يجب تحديد الإمتداد من خلال extension من object.
- بدلا من إستخدام store يجب إستخدام storeAs حيث يجب تمرير (المسار ، إسم الملف الذي نريد مع الإمتداد).

ماذا لو كنا لا نريد أن يتم تخزين إسم المجلد في قاعدة البيانات، وأن يتم تخزين فقط إسم الصورة

- إستخدام دالة getClientOriginalName بدلا من extension.
- إستخدام saveAs ومن ثم تمرير (المجلد ، إسم المف الذي تم الحصول عليه من الخطوة السابقة).

لماذا يتم رفع الصوره بداخل app/storage وكيف أتحكم بذلك
الملف الذي يتحكم بمسار رفع الملفات هو config/filesystems.php حيث يحتوي على ما يلي:

- المسار driver الإفتراضي وهو local.

```
'default' => env('FILESYSTEM_DRIVER', 'local'),
```

- بداخل المصفوفة disks نجد مجموعة من drivers وهي local, public, s3، فإذا أردنا أن يتم تغيير المسار الإفتراضي نقوم بتغيير الـ FILESYSTEM_DRIVER إلى s3 أو public.
- كما نلاحظ أنه تم إضافة خيار أخر وهو 'public' إلى دالة storeAs.
- إلا أنه حتى بعد إستخدام الـ public disk نرى أن المجلد هو storage بمعنى أن الملفات لن يتم عرضها، **ولحل هذه المشكله يمكن إستخدام أحد الطرق التالية:-**
- **الطريقة الأولى** : إنشاء إختصار من مجلد storage بداخل مجلد public ويتم ذلك من خلال تنفيذ الأمر

```
php artisan storage:link
```

- حيث ينشئ الأمر sim-link(shortcut) باسم storage بداخل المجلد public

```
App/public/storage
```

- بذلك تصبح هذه الصور والملفات public.
- **الطريقة الثانية :** التعديل في public disk مباشرة،

```
'public' => [
    'driver' => 'local',
    'root' => public_path('uploads'),
    'visibility' => 'public',
],
```

- كما نرى أنه تم تغيير storage_path إلى public_path ومن ثم إسم المجلد، وإزالة url.
- بالتالي سيتم إنشاء مجلد باسم uploads بداخل app/public وسيتم تخزين الصور بداخلة.
-
- **التحقق من الصور/ الملفات Validation**

للتحقق أن قيمة avatar هي ملف

```
$request->validate([
    'avatar'=>'file',
]);
```

- **جعل الحقل إجباري required و التحقق من نوع الملف يتم إضافة mimes ، والتحقق من الحجم الأقصى للملف max**

```
$req->validate([
    'file' => 'required|mimes:csv,txt,xlx,xls,pdf|max:2048'
]);
```

- **التحقق أن ما يتم رفعة صور فقط**

```
$request->validate([
    'avatar'=>'image',
```

```
]);
```

- بالوضع الإفتراضي هنا سيتم السماح بالإمتدادات التالية (jpg, jpeg, png, bmp, gif, svg, webp).
- ولتحديد إمتدادات معينة يمكن إستخدام mimes

```
$request->validate([
    'avatar'=>'image|png',
]);
```

- **لتحديد الحجم المسموح**

```
$request->validate([
    'avatar'=>'image|size:30',
]);
```

- هنا تم تحديد الحجم المسموح بـ 30kb.

- **لتحديد أبعاد الصور**

```
$request->validate([
    'avatar'=>'image|dimensions:min_width=200,min_height:200',
]);
```

- **تحديد أبعاد الصورة من خلال نسبة وعرض الصورة ratio**

```
$request->validate([
    'avatar'=>'image|dimensions:ration=3/2',
]);
```

- **كيف يمكن التعديل على حجم وأبعاد الصور أثناء الرفع**
- بداية قبل البدء سأقوم بإرجاع قيم disc public في ملف filesystems.php إلى القيم التالية

```
'default' => env('FILESYSTEM_DRIVER', 'public'),

'public' => [
    'driver' => 'local',
    'root' => storage_path('app/public'),
    'url' => env('APP_URL').'/storage',
    'visibility' => 'public',
```

- ببعض الأحيان ومن أجل سرعة الموقع نحتاج إلى صور مصغره عن الصوره الأصلية ، وللقيام بذلك يمكن إستخدام intervention/image وهي حزمة PHP وليس خاصه بـ Laravel

```
Composer require intervention/image
```

- كود الرفع وإنشاء صوره مصغره

```
if($request->hasFile('avatar')){
    $file=$request->file('avatar');
    $filename=$file->getClientOriginalName();
    $file->storeAs('avatars/'.$author->id,$filename);
    $image=Image::make(storage_path('app/public/avatars/'.$author->id.'/'.$filename));
    $image->resize(50,50);
    $image->save(storage_path('app/public/avatars/'.$author->id.'/thumb-'.$filename));

  $author->update([
      'avatar'=>$filename
```

```
    ]);

}
```

- هنا نلاحظ أنه تم إنشاء صوره أخرى مصغره وتخزينها باسم thumb-$filename، لكن المشكله هناك أنه تم تحديد طول وعرض الصوره، بغض النظر عن أن الصورة مربعة أو مستطيله مما يتسبب بمشكله في شكل الصوره المصغره حيث يجب أن يتم التصغير حسب النسبة
- يتم التصغير بإستخدام بالنسبة (نسبة العرض الى نسبة الطول) من خلال إستبدال دالة resize ب fit.

```
$image=Image::make(storage_path('app/public/avatars/'.$author->id.'/'.$filename));

$image->fit(50,50);//change resize to fit

$image->save(storage_path('app/public/avatars/'.$author->id.'/thumb-'.$filename));

```

- هنا يمكننا أيضا عمل refactor للكود بحيث يصبح بالشكل التالي

```
Image::make(storage_path('app/public/avatars/'.$author->id.'/'.$filename))

    ->fit(50,50)

  ->save(storage_path('app/public/avatars/'.$author->id.'/thumb-'.$filename));
```

- **التعديل على ملف php.ini لحجم الملفات.**
- بالوضع الإفتراضي في php يجب أن يكون حجم الملف ليس أكبر من 2MB، فإذا أردنا زيادة الحجم يجب تعديل قيمة upload_max_filesize، وكذلك يجب تعديل قيمة Post_max_size

- Php.ini default

```
Upload_max_filesize=2M

Post_max_size=8M
```

- Change to

- Upload_max_filesize=20M
- Post_max_size=21M

- كيف يتم عرض الصور

-