Gestion des fichiers

1. Introduction :

Laravel utilise Flysystem comme composant de gestion de fichiers. Il en propose une API bien pensée et facile à utiliser. On peut ainsi manipuler fichiers et dossiers de la même manière en local ou à distance, que ce soit en FTP ou sur le cloud.

La configuration du système se trouve dans le fichier **config/filesystem.php**. Par défaut on est en local :

'default' => env('FILESYSTEM_DISK', 'local'),

Mais on peut aussi choisir FTP, SFTP ou s3.

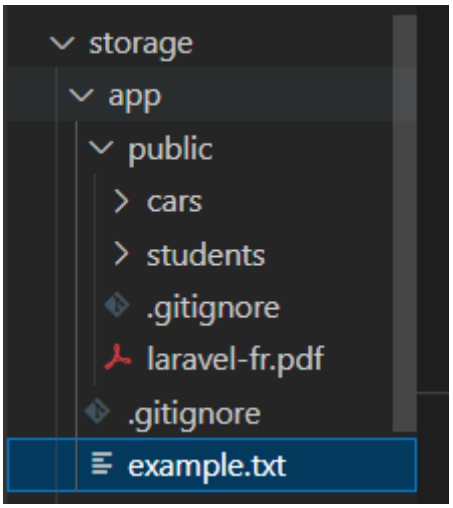
On peut définir des « disques » (**disks**) qui sont des cibles combinant un driver, un dossier racine et différents éléments de configuration :

2. Le disque local

Lorsque vous utilisez le pilote local, toutes les opérations sur les fichiers sont relatives au répertoire racine défini dans votre fichier de configuration des systèmes de fichiers. Par défaut, cette valeur est fixée au répertoire storage/app. Par conséquent, la méthode suivante écrirait dans le répertoire storage/app/example.txt :

```
use Illuminate\Support\Facades\Storage;


Storage::disk('local')->put('example.txt', 'Contents');
```



3. Le disque public

Le disque public inclus dans le fichier de configuration du système de fichiers de votre application est destiné aux fichiers qui seront accessibles au public. Par défaut, le disque public utilise le pilote local et stocke ses fichiers dans storage/app/public.

Pour que ces fichiers soient accessibles depuis le web, vous devez créer un lien symbolique entre public/storage et storage/app/public. L'utilisation de cette convention de dossier permet de conserver vos fichiers accessibles au public dans un seul répertoire qui peut être facilement partagé entre les déploiements lors de l'utilisation de systèmes de déploiement sans interruption.

Pour créer le lien symbolique, vous pouvez utiliser la commande Artisan :

```
php artisan storage:link
```

Une fois qu'un fichier a été stocké et que le lien symbolique a été créé, vous pouvez créer une URL vers les fichiers à l'aide de l'assistant d'actifs :

```
<div id="app">{{asset('/storage/test.txt')}}</div>
```

4. Obtention d'instances de disque

La façade **Stockage** peut être utilisée pour interagir avec n'importe lequel des disques configurés. Par exemple, vous pouvez utiliser la méthode **put** de la façade pour stocker un avatar sur le disque par défaut. Si vous appelez des méthodes de la façade **Storage** sans appeler au préalable la méthode **disk**, la méthode sera automatiquement transmise au disque par défaut :

```
Storage::disk('public')->put('avatars/1', $content);
```

5. Récupération du contenu d'un fichier

La méthode **get** peut être utilisée pour récupérer le contenu d'un fichier. La méthode renvoie la chaîne de caractères brute du fichier. N'oubliez pas que tous les chemins d'accès aux fichiers doivent être spécifiés par rapport à l'emplacement "racine" du disque :

```
  if (Storage::disk('local')->exists('/public/avatars/1')) {
    $content= Storage::get('/public/avatars/1');
      echo $content;
   }
```

5.1 Téléchargement de fichiers

La méthode **download** peut être utilisée pour générer une réponse qui force le navigateur de l'utilisateur à télécharger le fichier au chemin donné. La méthode **download** accepte un nom de fichier comme deuxième argument de la méthode, qui déterminera le nom de fichier qui sera vu par l'utilisateur téléchargeant le fichier. Enfin, vous pouvez passer un tableau d'en-têtes HTTP comme troisième argument de la méthode :

```
return Storage::download('public/laravel-fr.pdf',$file_name);
```

6. Stockage des fichiers

La méthode **put** peut être utilisée pour stocker le contenu d'un fichier sur un disque. Vous pouvez également passer une ressource PHP à la méthode **put**, qui utilisera le support de flux sous-jacent de Flysystem. N'oubliez pas que tous les chemins d'accès aux fichiers doivent être spécifiés par rapport à l'emplacement "root" configuré pour le disque :

```
use Illuminate\Support\Facades\Storage;


Storage::put('file.jpg', $contents);

Storage::disk('public')->put('example.txt', "contents");
```

6.1 Chargements de fichiers (upload)

Dans les applications web, l'un des cas d'utilisation les plus courants pour le stockage de fichiers est le stockage de fichiers téléchargés par l'utilisateur, tels que des photos et des documents. Laravel facilite le stockage des fichiers téléchargés en utilisant la méthode **store** sur une instance de fichier téléchargé. Appelez la méthode **store** avec le chemin où vous souhaitez stocker le fichier téléchargé :

```
public function update(Request $request): string
  {
      $path = $request->file('avatar')->store('avatars');

      $path = $request->file('avatar')->store('avatars', 'public');


      return $path;
  }
```

Il y a quelques points importants à noter dans cet exemple. Nous n'avons spécifié qu'un nom de répertoire, et non un nom de fichier. Par défaut, la méthode store génère un identifiant unique qui sert de nom de fichier. L'extension du fichier sera

déterminée en examinant le type MIME du fichier. Le chemin d'accès au fichier sera renvoyé par la méthode store afin que vous puissiez stocker le chemin d'accès, y compris le nom de fichier généré, dans votre base de données.

7. Supprimer un fichier

```
use Illuminate\Support\Facades\Storage;

Storage::delete('file.jpg');

Storage::delete(['file1.jpg', 'file2.jpg']);
```

Si nécessaire, vous pouvez spécifier le disque à partir duquel le fichier doit être supprimé :

```
use Illuminate\Support\Facades\Storage;


Storage::disk('public')->delete('path/file.jpg');
```