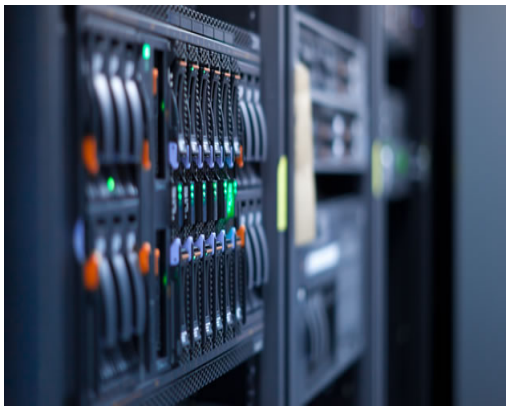
Bagaimana kami merancang struktur folder storage untuk mempermudah backup file secara berkala.



Oleh : Hasnul Arief Fikri

28 Maret 2025

Sumber : https://fiona.usu.ac.id/bagaimana-kami-merancang-struktur-folder-storage-untuk-mempermudah-backup-file-secara-berkala

Disclaimer: Tulisan ini merupakan pengalaman penulis sebagai konsultan IT di luar USU.

Di awal pengembangan sistem informasi kami selalu berfokus dengan skema database untuk mengoptimasi setiap query dapat berjalan optimal dan seamless. Dengan respon time dalam rentang millisecond, setiap request yang direspon dipastikan dapat di-deliver sesegera mungkin kepada user tanpa ada lag ataupun delay.

Berbeda dengan file storage, kami terbiasa tidak acuh dan hanya menempatkan file berdasarkan modul

Storage/kepegawaian/user_1/ktp.png

Atau

Storage/keuangan/pembayaran_001.png

Singkat cerita sistem berjalan bulan demi bulan dengan ratusan hingga ribuan file yang diupload dan tak terasa penggunaan storage sudah mencapai puluhan gigabyte. Mungkin bukan hal yang besar bagi sebagian pengembang namun bagi kami ini sudah merupakan prestasi mengingat klien kami hanyalah perusahan kelas menengah dengan kemampuan pengadaan/pembiayaan IT terutama infrastruktur yang tergolong kecil.

Dengan server tua, storage menggunakan harddisk jenis lama, dan tidak mendukung RAID, hal ini menjadi bom waktu bagi kami seandainya tiba-tiba server crash dan data corrupt.

Untuk database kami bisa gunakan mysql daily backup, dengan menggunakan remote mysql kita bisa melakukan dump sql setiap malam dari server ke komputer backup, *easy*. Lalu bagaimana dengan folder storage? Inilah yang menjadi masalah, storage yang telah digunakan puluhan GB, apakah setiap malam akan melakukan *zipping* dan mendownload arsip puluhan giga tersebut? Tentunya ini akan membebani server, jaringan, dan juga waktu pengerjaan. Sangat tidak efisien mengingat setiap hari mungkin data yang diupload hanya kisaran 10-50MB namun kita harus membackup seluruh folder storage.

Dengan struktur yang sudah didesain sejak dahulu kami kesulitan memilah mana data lama dan mana data yang baru sehingga harus backup seluruh folder storage. Akhirnya tim melakukan kordinasi dan memutuskan bahwa code harus direfactor dan storage harus direstruktur.

Sehubungan karena dukungan finansial yang tidak mungkin ada maka daily backup adalah satu-satunya solusi. Namun idenya adalah bagaimana agar daily backup tidak terasa berat dan tidak membebani dari sisi server, jaringan, dan juga waktu eksekusi.

Desain baru

Pada eksekusi file upload terbaru kami mengubah struktur menjadi

Storage/{Y-m}/kepegawaian/user_1/ktp.png

Atau

Storage/{Y-m}/keuangan/pembayaran_001.png

Dimana {Y-m} merupakan Tahun-Bulan file tersebut diupload. Jika file tersebut diupload pada bulan Maret 2025 maka storage tempat file tersebut disimpan menjadi **Storage/2025-03/path/to/file.ext**

Tujuannya apa? Pada bulan berjalan (misal saat ini bulan Maret 2025), kami hanya perlu membackup folder **2025-03** setiap malam dan jika sudah pindah bulan ke bulan April kami akan lanjut membackup **2025-04** setiap malam. Bagaimana dengan **2025-03** ? Kita sudah pastikan sebelumnya bahwa pada bulan April tidak akan ada file yg diupdate ke **2025-03** sehingga kita tidak perlu lagi membackup file tersebut setiap malam.

Bagaimana jika file yang ada pada 2025-03 kebetulan harus diupdate/diganti? Jika seperti itu file lama yang ada di 2025-03 akan dihapus dan file baru akan diletkaan di 2025-04. Contoh: Pada bulan maret user_1 mengupload KTP sehingga folder yang terbentuk adalah

Storage/2025-03/kepegawaian/user_1/ktp.png

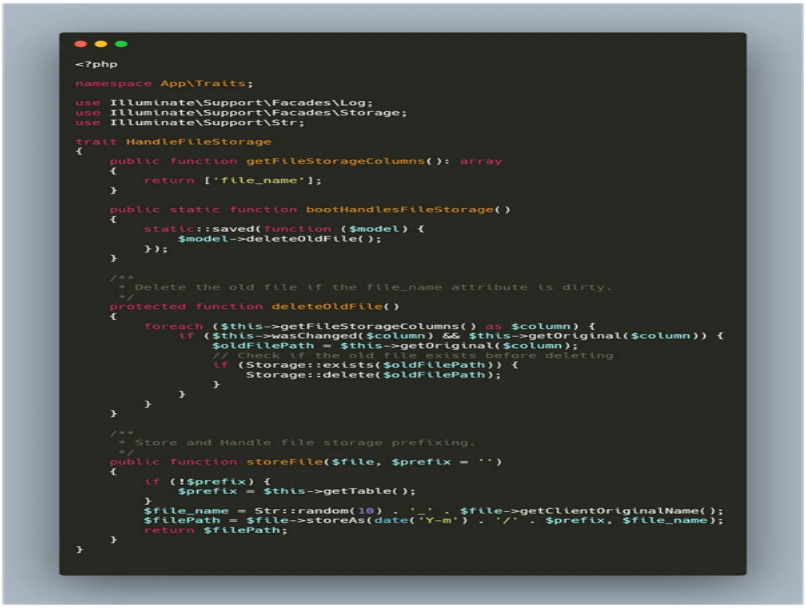
Pada bulan April user_1 mengganti file KTP nya sehingga yang terjadi

~~Storage/2025-03/kepegawaian/user_1/ktp.png~~ [dihapus]

Storage/2025-04/kepegawaian/user_1/ktp.png

Bagaimana dengan folder 2025-03 yang sudah dibackup? Kan ada perubahan? Ya tidak masalah kita anggap itu sebagai resiko storage.

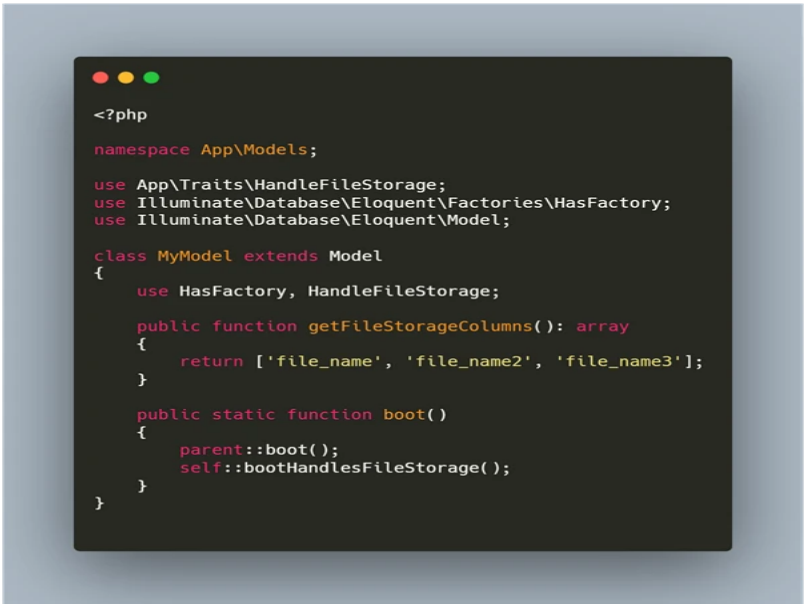
Untuk implementasi hal tersebut kami mencoba membuat Trait yang dapat digunakan oleh semua Model kerja kami di project Laravel



```php
<?php

namespace App\Traits;

use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Str;

trait HandleFileStorage
{
    public function getFileStorageColumns(): array
    {
        return ['file_name'];
    }

    public static function bootHandlesFileStorage()
    {
        static::saved(function ($model) {
            $model->deleteOldFile();
        });
    }

    /**
     * Delete the old file if the file_name attribute is dirty.
     */
    protected function deleteOldFile()
    {
        foreach ($this->getFileStorageColumns() as $column) {
            if ($this->wasChanged($column) && $this->getOriginal($column)) {
                $oldFilePath = $this->getOriginal($column);
                // Check if the old file exists before deleting
                if (Storage::exists($oldFilePath)) {
                    Storage::delete($oldFilePath);
                }
            }
        }
    }

    /**
     * Store and Handle file storage prefixing.
     */
    public function storeFile($file, $prefix = '')
    {
        if (!$prefix) {
            $prefix = $this->getTable();
        }
        $file_name = Str::random(10) . '_' . $file->getClientOriginalName();
        $filePath = $file->storeAs(date('Y-m') . '/' . $prefix, $file_name);
        return $filePath;
    }
}
```

Secara default kami menggunakan nama kolom “file_name” di dalam Model kami. Jika nama kolom berbeda dan mungkin dalam satu model terdapat lebih dari satu kolom tempat menampung nama file maka fungsi **getFileStorageColumns** bisa dioverride di Model.

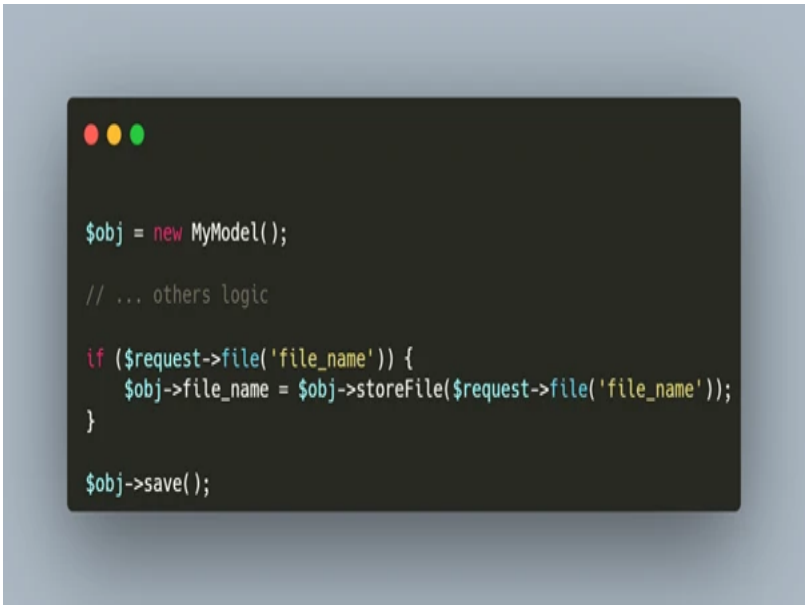
Untuk di Model penggunaannya kira-kira sebagai berikut



```php
<?php

namespace App\Models;

use App\Traits\HandleFileStorage;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class MyModel extends Model
{
    use HasFactory, HandleFileStorage;

    public function getFileStorageColumns(): array
    {
        return ['file_name', 'file_name2', 'file_name3'];
    }

    public static function boot()
    {
        parent::boot();
        self::bootHandlesFileStorage();
    }
}
```

Kami meregister observer untuk mentrigger penghapusan file lama jika model disimpan.

Kemudian untuk controller



```php
$obj = new MyModel();

// ... others logic

if ($request->file('file_name')) {
    $obj->file_name = $obj->storeFile($request->file('file_name'));
}

$obj->save();
```

Dengan begini setiap kami mengeksekusi **storeFile(),** folder penyimpanan akan dihandle oleh Trait dan *fullpath* akan direturn dan dapat disimpan pada kolom **file_name**. Begitu juga dengan file lama yang akan dihapus saat model disimpan **$obj->save()**

Demikian *approach* yang kami buat, belum selesai, masih ada yang harus dikembangkan seperti:

- Penghapusan file saat **$obj->delete()** dan bagaimana perlakuannya dengan softdelete model,
- Daily backup command, mungkin akan saya buat pada tulisan berikutnya.

Salam Programmer ASN