Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : FILE UPLOAD

Mata Kuliah Praktikum Pemrograman Web Lanjut (PWL)

Mei 2024

Nama : Elvaretta Salsabilla

NIM : 2241720116

No/Kelas : 14/TI-2H

JOBSHEET 12

FILE UPLOAD

A. PERSIAPAN AWAL

1. Membuat beberapa file yang diperlukan, yang pertama controller :
   php artisan make:controller FileUploadController

```
PS C:\laragon\www\PWL_POS> php artisan make:controller FileUploadController

   INFO  Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\FileUploadCon
troller.php] created successfully.
```

2. Buat 2 buah method yaitu method fileUpload() dan prosesFileUpload. Method fileUpload() berisi kode program yang akan memanggil file view file-upload. blade.php. File blade inilah yang berisi kode HTML dan CSS untuk membuat form (akan kita buat sesaat lagi). Sedangkan method prosesFileUpload() untuk memproses hasil submit form. kita akan banyak membuat kode program, diantaranya validasi form dan serta proses pemindahan file yang sudah di upload. Sebagai argument terdapat variabel $request yang akan berisi Request object.

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

0 references | 0 implementations
class FileUploadController extends Controller
{
    0 references | 0 overrides
    public function fileUpload(){
        return view('file-upload');
    }

    0 references | 0 overrides
    public function prosesFileUpload(Request $request){
        return "Pemrosesan file upload disini";
    }
}
```

3. Membuat route pada web.php

```php
Route::post('/file-upload', [FileUploadController::class , 'fileUpload']);
Route::get('/file-upload', [FileUploadController::class , 'prosesfileUpload']);
```

Route pertama dipakai untuk menampilkan form dengan cara mengakses method fileUpload() yang ada di FileUploadController. Alamat URLnya adalah '/file-upload'. Sedangkan route kedua berfungsi untuk pemrosesan form dengan mengakses method prosesfileUpload() di FileUploadController. Route ini menggunakan method post karena menjadi tujuan saat form di submit.

4. Membuat File View file-upload.blade.php dengan kode sebagai berikut :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
    rel="stylesheet"
          integrity="sha384-OMT1Xy2jpepJSiVsWAUR0PoRePkoC6YtYmDp5plYj12Bxj1hoWmjY6hvhHAElw6h"
          crossorigin="anonymous">
    <title>File Upload</title>
</head>
<body>
    <div class="container mt-3">
        <h2>File Upload</h2>
        <hr>

        <form action="{{ url('/file-upload') }}" method="POST" enctype="multipart/form-data">
            @csrf
            <div class="mb-3">
                <label for="berkas" class="form-label">Gambar Profile</label>
                <input type="file" class="form-control" id="berkas" name="berkas">
                @error('berkas')
                    <div class="text-danger">{{ $message }}</div>
                @enderror
            </div>
            <button type="submit" class="btn btn-primary my-2">Upload</button>
        </form>
    </div>
</body>
</html>
```

Gunakan CDN Bootstrap : https://getbootstrap.com/ agar tampilan lebih menarik

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
    crossorigin="anonymous">
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
    crossorigin="anonymous">
</script>
```

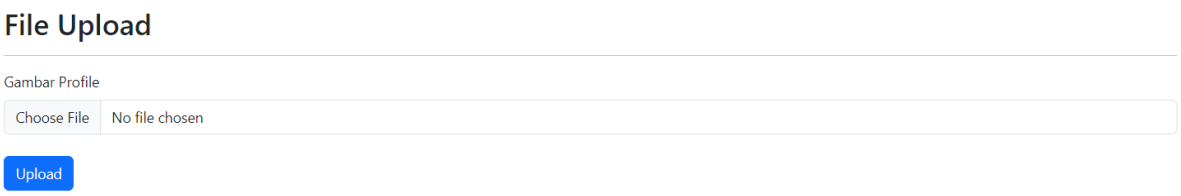
Sehingga tampilan menjadi seperti berikut :

Jurusan Teknologi Informasi Politeknik Negeri Malang

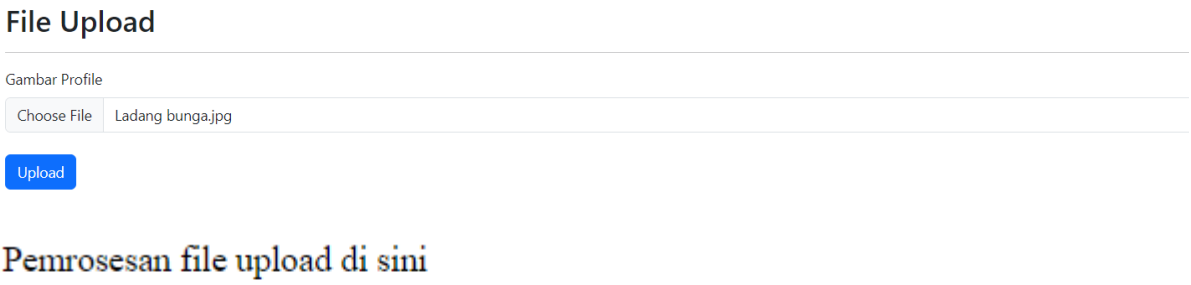
Jobsheet-12 : FILE UPLOAD

Mata Kuliah Praktikum Pemrograman Web Lanjut (PWL)

Mei 2024



Untuk membuat tampilan form, menggunakan sedikit class CSS bawaan Bootstrap (Gunakan CDN Bootstrap). Karena kita akan mengupload file, maka di dalam tag harus ditambah atribut enctype="multipart/form-data" seperti di baris 20. Form ini dikirim menggunakan method POST ke url('/file-upload'). Seperti biasa, form yang dikirim menggunakan method post juga harus menyertakan perintah @csrf. Kode ini ada di baris 21. Form ini hanya terdiri dari 1 inputan, . Kemudian terdapat blok @error('berkas') untuk menampilkan pesan error validasi di baris 25-26. Terakhir, tombol submit isi dengan nama "Upload". Langsung saja kita coba test. Silahkan upload sembarang file, lalu klik tombol "Upload":



B. INFORMASI FILE UPLOAD

1. Informasi seputar file yang sudah di upload bisa kita akses melalui Request object, yakni variabel $request. Silahkan isi kode berikut ke dalam method prosesfileUpload() di FileUploadController:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

3 references | 0 implementations
class FileUploadController extends Controller
{
    1 reference | 0 overrides
    public function fileUpload()
    {
        return view('file-upload');
    }
    1 reference | 0 overrides
    public function prosesFileUpload(Request $request)
    {
        dump($request->berkas);
        //dump($request->file('file'));
        //return "Pemrosesan file upload di sini";
    }
}
```

Isi dari method ini hanya 1 baris, langsung men-dump $request->berkas. Nama "berkas" ini berasal dari nilai artibut name pada tag <input type=”file” name=”berkas”>.

2. lakukan 2 percobaan, pertama langsung submit form tanpa mengisi file apapun. Dan kedua, upload file sembarang dan submit. Screen Shot bagaimana hasilnya?
   - Upload form kosong

```
null // app\Http\Controllers\FileUploadController.php:15
```

   - Upload gambar

```
Illuminate\Http\UploadedFile {#1445 ▼ // app\Http\Controllers\FileUploadController.php:15
  -test: false
  -originalName: "Ladang bunga.jpg"
  -mimeType: "image/jpeg"
  -error: 0
  #hashName: null
  path: "C:\Users\elva\AppData\Local\Temp"
  filename: "php8948.tmp"
  basename: "php8948.tmp"
  pathname: "C:\Users\elva\AppData\Local\Temp\php8948.tmp"
  extension: "tmp"
  realPath: "C:\Users…\Temp\php8948.tmp"
  aTime: 2024-05-29 07:38:24
  mTime: 2024-05-29 07:38:24
  cTime: 2024-05-29 07:38:24
  inode: 5348024558104214
  size: 117370
  perms: 0100666
  owner: 0
  group: 0
  type: "file"
  writable: true
  readable: true
  executable: false
  file: true
  dir: false
  link: false
  linkTarget: "C:\Users\elva\AppData\Local\Temp\php8948.tmp"
}
```

Jika tidak ada file yang di upload, perintah $request->berkas akan mengembalikan nilai NULL. Namun jika ada file yang di upload, akan menampilkan beragam informasi mengenai file tersebut. Diantaranya nama file, mimetype, tanggal upload serta ukuran file yang di upload. Untuk memeriksa apakah terdapat sebuah file yang di upload, bisa menggunakan perintah $request->hasFile('berkas'). Hasilnya true jika ada file yang di upload dan false jika tidak ada file yang di upload.

3. Berikut cara mengambil beberapa info dari file yang di upload: (app/Http/Controllers/FileUploadController.php)

```php
class FileUploadController extends Controller
{
    1 reference | 0 overrides
    public function fileUpload()
    {
        return view('file-upload');
    }
    1 reference | 0 overrides
    public function prosesFileUpload(Request $request)
    {
        // ?
        //dump($request->file('file'));
        //return "Pemrosesan file upload di sini";

        if ($request->hasFile('berkas')) {
            echo "path(): " . $request->berkas->path();
            echo "<br>";
            echo "extension(): " . $request->berkas->extension();
            echo "<br>";
            echo "getClientOriginalExtension(): " .
            $request->berkas->getClientOriginalExtension();
            echo "<br>";
            echo "getMimeType(): " . $request->berkas->getMimeType();
            echo "<br>";
            echo "getClientOriginalName(): " . $request->berkas->getClientOriginalName();
            echo "<br>";
            echo "getSize(): " . $request->berkas->getSize();
        } else {
            echo "Tidak ada berkas yang diupload";
        }
    }
}
```

4. Lakukan percobaan upload file sehingga akan muncul keterangan sbb:

```
path(): C:\Users\elva\AppData\Local\Temp\phpD4FF.tmp
extension(): jpg
getClientOriginalExtension(): jpg
getMimeType(): image/jpeg
getClientOriginalName(): Ladang bunga.jpg
getSize(): 117370
```

Penjelasan :

Di baris 18 pada method prosesFileUpload terdapat sebuah kondisi if yang akan dijalankan jika $request->hasFile('berkas') bernilai true. Di dalam blok ini, untukmengakses beberapa method. File sample yang diupload adalah sebuah file excel . Berikut penjelasan dari method prosesFileUpload yang ada di baris 20 - 32:

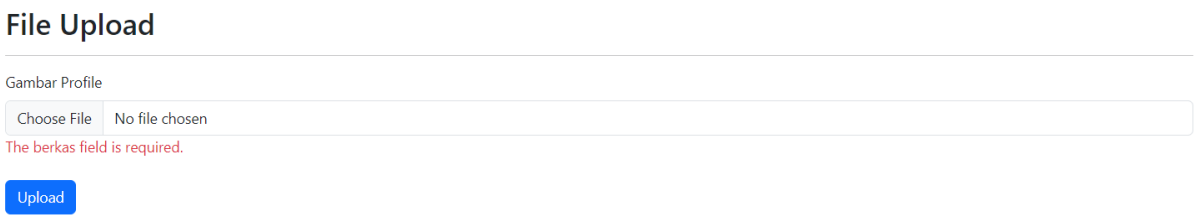
- $request->berkas->path(), menampilkan alamat path dari file yang sudah di upload.Perhatikan bahwa isinya adalah alamat temporary dari pengaturan PHP. Dalam contoh ini, alamat file yang diupload akan disimpan di C:\Users\Dimas_Polinema\AppData\Local\Temp\php5674.tmp.
- $request->berkas->extension(), menampilkan extension file. Dalam contoh ini file yang diupload file excel, sehingga extensionnya adalah xls.
- $request->berkas->getClientOriginalExtension(), menampilkan extension yang diambil dari nama file. Karena file yang di upload adalah excel, maka hasil method ini adalah xls.
- $request->berkas->getMimeType(), menampilkan mimetype dari file yang di upload.Dalam contoh ini hasilnya excel. Mimetype sendiri adalah sejenis standar daftar jenis file yang ditulis dalam format "type/subtype". Dalam hal ini, file excel bertipe "application", dan subtype "vnd.ms-excel". Lebih lanjut tentang mimetype bisa dibaca-baca ke: MIME types (IANA media types).
- $request->berkas->getClientOriginalName(), menampilkan nama asli dari file yang diupload, yang dalam contoh ini berupa " FormatNilai-DIV2C_SIB - 2C-Pemrograman_Web.xls ".
- $request->berkas->getSize(), menampilkan ukuran file yang di upload (dalam satuan byte). Dalam contoh ini, file excel yang diupload berukuran 845312 byte

C. VALIDASI FILE UPLOAD

1. Membuat validasi file upload mirip seperti cara membuat validasi inputan form biasa, yakni menggunakan method $request->validate(). Berikut contoh penggunaannya: (app/Http/Controllers/FileUploadController.php)

```
$request->validate([
    'berkas'=>'required',
]);
echo $request->berkas->getClientOriginalName()."lolos
validasi";
```

Terdapat syarat required untuk file berkas, sehingga akan tampil pesan error jika form di submit tanpa meng-upload sebuah file:



Jika syarat validasi tidak terpenuhi, akan langsung di redirect untuk menampilkan pesan error. Di halaman view file-upload.blade.php kita sudah menyiapkan perintah @error('berkas') untuk menampilkan pesan error ini

2. Syarat validasi selanjutnya adalah membatasi jenis file dan maksimal ukuran file : (tambahkan code pada line 20)

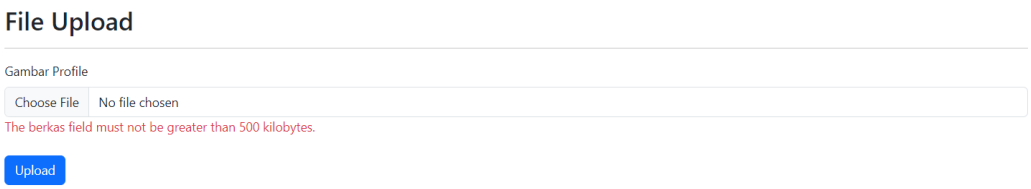
```
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
echo $request->berkas->getClientOriginalName()."lolos
validasi";
```

Berikut ini penjelasannya : Berikut penjelasan dari syarat validasi ini:

- required: inputan form tidak boleh kosong.
- file: memastikan bahwa file sudah berhasil di upload.
- image: file yang di upload harus file image (gambar), salah satu dari jpeg, png, bmp, gif, svg, atau webp.
- max:5000, artinya file yang di upload berukuran maksimal 5000 byte atau sekitar 5 MB.

Yang perlu sedikit penjelasan adalah tentang syarat max:5000. Meskipun ini artinya file dengan ukuran 5MB bisa di upload, tapi ini masih dibatasi oleh pengaturan upload_max_filesize di file php.ini.

Hasilnya:



D. MEMINDAH FILE UPLOAD

Semua file yang di upload akan tersimpan sementara ke folder temporary yang dalam PHP bawaan XAMPP berada di C:\xampp\tmp\. Agar bisa diakses, file ini harus di pindah ke folder aplikasi kita. Laravel menyediakan beragam cara untuk memindahkan file ini, diantaranya method store(), storeAs(), dan move(). Kita akan bahas secara bertahap.

1. Cara pertama adalah menggunakan method store() yang bisa diakses dari Request object. Berikut contoh penggunaannya:

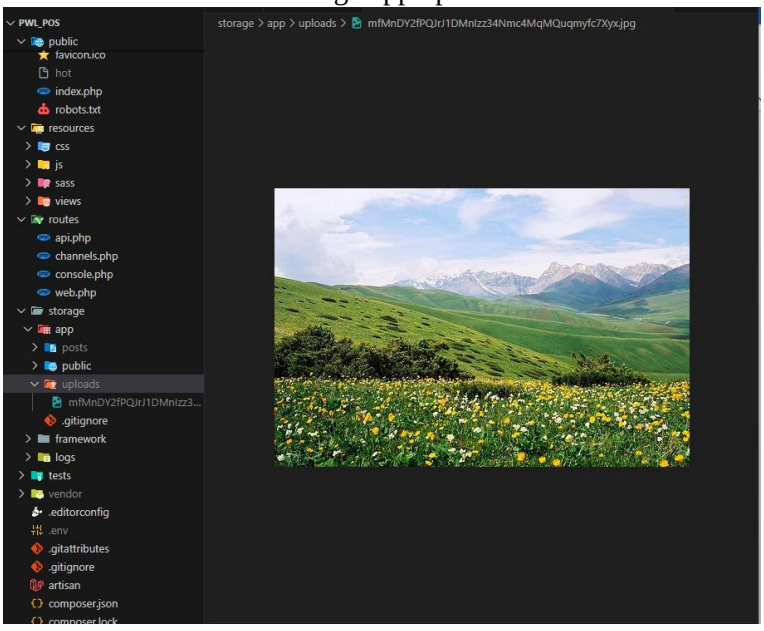
```
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
$path = $request->berkas->store('uploads');
echo "proses upload berhasil, file berada di: $path";
```

Di baris 17-20 terdapat kode untuk proses validasi yang sudah di bahas sebelumnya. Proses pemindahan file dilakukan di baris 19 dengan perintah $request->berkas->store('uploads'). Method store() akan mengambil file upload dari folder temporary dan memindahkannya ke folder storage\app\ di aplikasi Laravel. Method store() ini butuh sebuah argument berupa nama folder, sehingga perintah $request>berkas->store('uploads') akan memindahkan file upload ke folder storage\app\uploads.Nilai kembalian dari method ini berupa alamat path dari file akhir, yang dalam contoh ini disimpan ke dalam variabel $path.

Mari kita coba, silahkan upload sebuah file gambar dengan ukuran kurang dari 1MB:

```
proses upload berhasil, file berada di: uploads/mfMnDY2fPQJrJ1DMnIzz34Nmc4MqMQuqmyfc7Xyx.jpg
```

Setelah itu buka folder storage\app\uploads\ untuk melihat file ini:



file yang di upload sudah bisa diakses. Namun kenapa nama file acak seperti itu?

Method store() memang akan men-generate nama acak untuk setiap file yang di upload. Kesannya memang agak aneh, tapi sebenarnya sangat bermanfaat:

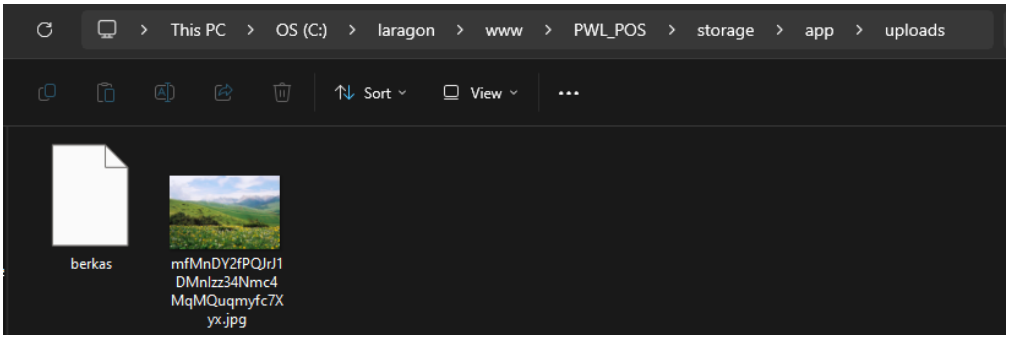
- Menghindari kemungkinan file ditimpa. Jika nama file yang sudah di upload sama dengan file asal, bisa saja di kemudian hari ada yang mengupload file dengan nama yang sama. Jika ini terjadi, maka file yang baru akan menimpa file lama.
- Menghindari error karena nama file mengandung spasi. Di dalam penulisan alamat path, karakter spasi ini sering menjadi masalah.

jika kita tetap ingin membuat nama file sendiri? Tidak masalah, Laravel menyediakan method storeAs() untuk keperluan ini. Berikut contoh penggunaannya: (line 20)

```
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
$path = $request->berkas->storeAs('uploads', 'berkas');
echo "proses upload berhasil, file berada di: $path";
```

Method storeAs() butuh 2 argument. Argument pertama berupa nama folder, dan argument kedua berupa nama file yang ingin di buat. Jika kita meng-upload file dengan perintah di atas, nama file akhir adalah "berkas", dan tanpa extension!

```
proses upload berhasil, file berada di: uploads/berkas
```



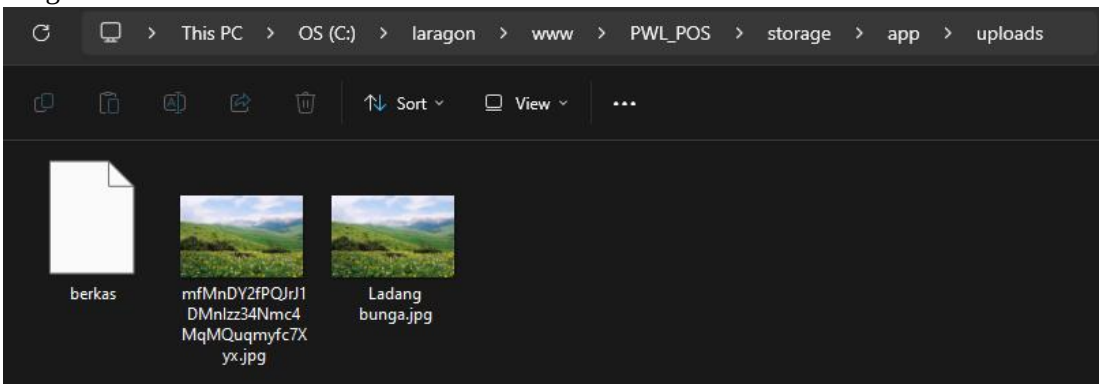
Sehingga kita harus menambah sendiri extension file ini. Selain itu, nama file tidak bisa di tulis langsung seperti ini karena akan terus tertimpa oleh file lain karena sama-sama bernama "berkas".

2. mengambil nama file dari fungsi getClientOriginalName() seperti contoh berikut: (app/Http/Controllers/FileUploadController.php)

```
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
$namaFile = $request->berkas->getClientOriginalName();
$path = $request->berkas->storeAs('uploads', $namaFile);
echo "proses upload berhasil, file berada di: $path";
```

Di baris 19 mengambil nama file asal dengan perintah $request->berkas>getClientOriginalName(), yang hasilnya ditampung oleh variabel $namaFile.Variabel $namaFile ini kemudian diinput sebagai argument kedua dari method

storeAs(). Dengan cara ini maka file yang di upload akan memiliki nama yang sama dengan file asal:

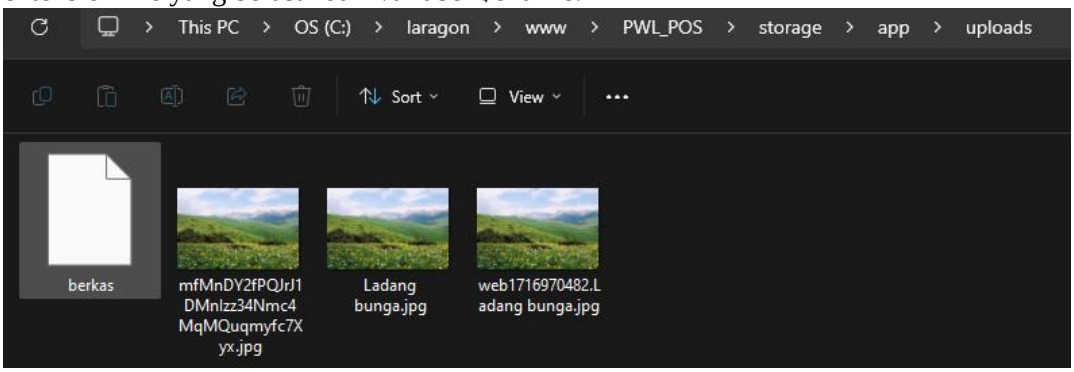


Namun terdapat kelemahan dengan teknik seperti ini. Jika ada yang meng-upload file dengan nama yang sama, maka file pertama akan tertimpa. Selain itu bisa timbul masalah jika nama file yang di upload mengandung karakter spasi.

3. Cara lain adalah dengan meng-generate nama acak sendiri. Sebagai contoh, membuat nama file upload berdasarkan nama user, ditambah dengan digit acak dari fungsi time(). Berikut kode programnya: (app/Http/Controllers/FileUploadController.php)

```
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
$textFile = $request->berkas->getClientOriginalName();
$namaFile ='web'.time()."_".$textFile;
$path = $request->berkas->storeAs('uploads', $namaFile);
echo "proses upload berhasil, data disimpan pada: $path";
```

Di baris19, method $request->berkas->getClientOriginalExtension() di pakai untuk mengambil extension file asal. Kemudian di baris 20 menyambung 3 string, yakni web-' yang diibaratkan sebagai nama user, hasil timestamp dari fungsi time() bawaan PHP, serta extension file yang berasal dari variabel $extFile.



Hasilnya, file yang di upload bernama web-1574953613.jpg. Nama user "web" ini nantinya bisa berasal dari database, yang akan berbeda-beda sesuai dengan id login. Nama ini relatif tidak bermasalah, kecuali user tersebut meng-upload beberapa file dalam

detik yang sama. Agar lebih aman lagi (supaya tidak ada nama file yang bentrok), bisa ditambah dengan karakter acak lain seperti hasil fungsi hash md5(). Atau daripada repot, bisa pakai method store() saja supaya Laravel yang langsung menggenerate nama file secara otomatis.

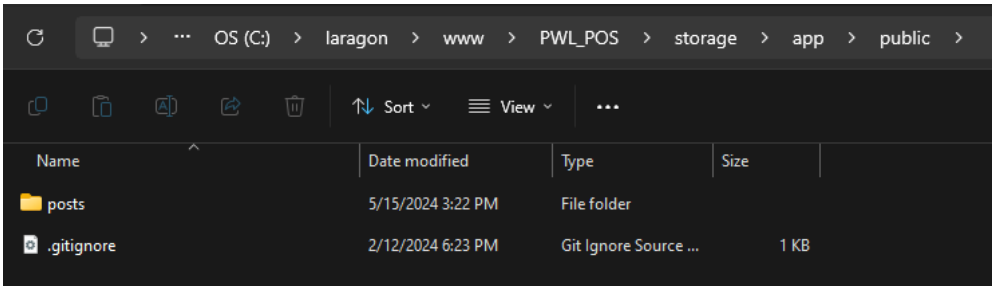
E. MEMBUAT SYMLINK

File yang kita upload sudah bisa diakses, tapi baru secara manual menggunakan Windows Explorer. File tersebut belum bisa dibuka dari halaman web karena secara default (dan memang sudah seharusnya), Laravel hanya mengizinkan web browser mengakses file yang ada di folder public saja. Sebenarnya bisa saja file upload langsung disimpan ke folder public, tapi Laravel memilih solusi yang lebih elegan memakai metode yang disebut sebagai symbolic link, atau sering disingkat sebagai symlink. Symlink mirip seperti shortcut tapi seolah-olah mengcopy isi satu folder ke folder lain (mirorring). Pada saat ini, semua file yang sudah di upload berada di folder storage\app\. Kita bisa membuat symlink agar file ini juga bisa diakses dari folder public\.

1. Membuat symlink yang menghubungkan folder storage\app dengan folder public\ php artisan storage:link

```
PS C:\laragon\www\PWL_POS> php artisan storage:link

  ERROR  The [C:\laragon\www\PWL_POS\public\storage] link already exists.
```

Dengan perintah ini, akan tampil sebuah symlink bernama storage di folder public, silahkan buka folder ini:



Ketika di klik, terdapat file .gitignore di dalam symlink storage. Ini hanya file bantu yang berfungsi sebagai tanda agar isi folder storage tidak perlu di backup oleh aplikasi Git. Symlink storage pada dasarnya merupakan shortcut atau mirror dari folder storage\app\public.

Dengan kata lain, semua file yang akan kita simpan di storage\app\public, juga akan ada di folder public\storage. Untuk uji coba, silahkan copy sebuah file ke folder storage\app\public, maka file tersebut juga ada di folder public\storage. Dan ketika file itu di hapus, maka di folder lain akan ikut terhapus. Dan ini berlaku dua arah, jika file di folder public\storage di tambah atau di hapus, file yang ada di storage\app\public jika akan mengikuti.

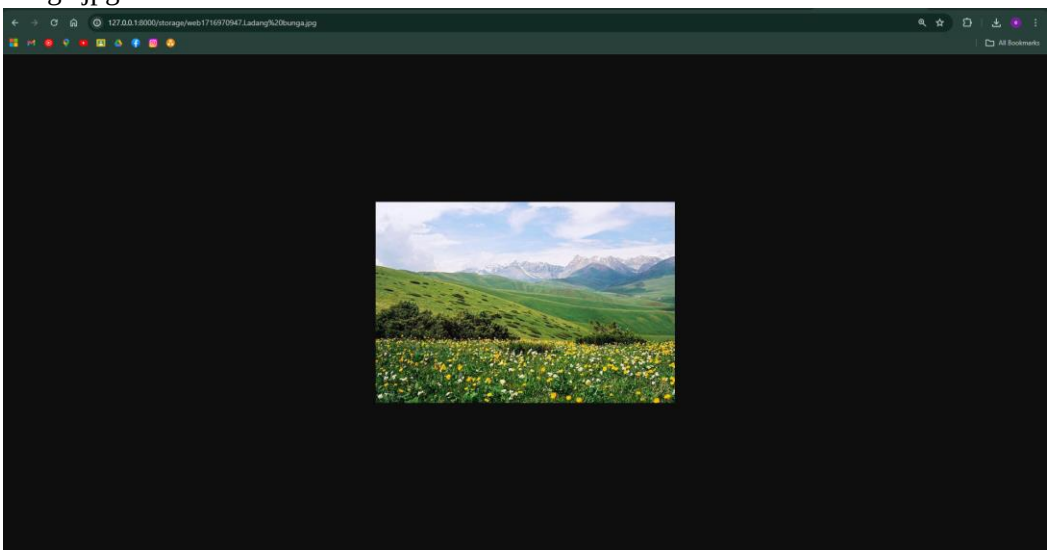
2. Modifikasi dari method prosesFileUpload():

```
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
$textFile = $request->berkas->getClientOriginalName();
$namaFile ='web'.time()."_".$textFile;
$path = $request->berkas->storeAs('public', $namaFile);
echo "proses upload berhasil, data disimpan pada: $path";
```

Perhatikan bahwa argumen pertama dari method storeAs() adalah 'public'(line 21). Inilah folder tempat file upload akan disimpan. Silahkan upload sebuah file, maka akan tampil hasil berikut:

```
proses upload berhasil, data disimpan pada: public/web1716970947.Ladang bunga.jpg
```

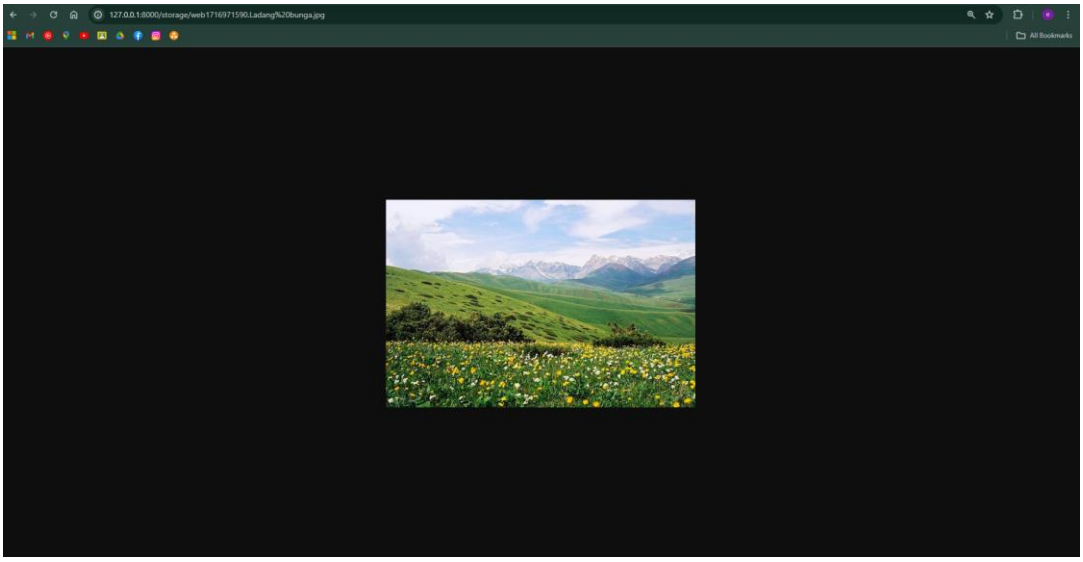
Proses upload berhasil, dan file tersebut ada di storage\app\public\ web1714801950.images.png. Nama file yang akan anda dapati pastinya berbeda karena di sini meng-generate nama file berdasarkan fungsi time(). Karena file upload sudah berada di dalam folder symlink, maka bisa diakses dari web browser. Silahkan buka alamat berikut (sesuaikan nama filenya): http://localhost:8000/storage/ web1716970947.Ladang bunga.jpg



3. Menambahkan link agar gambar dapat diakses, tambahkan kode pada method prosesFileUpload:

```php
$request->validate([
    'berkas'=>'required|file|image|max:500',
]);
$textFile = $request->berkas->getClientOriginalName();
$namaFile ='web'.time()."_".$textFile;
$path = $request->berkas->storeAs('public', $namaFile);

$pathBaru=asset('storage/'.$namaFile);
echo "proses upload berhasil, data disimpan pada:$path";
echo "<br>";
echo "Tampilkan link: <a href='$pathBaru'>$pathBaru</a>";
```

proses upload berhasil, data disimpan pada:public/web1716971590.Ladang bunga.jpg
Tampilkan link: http://127.0.0.1:8000/storage/web1716971590.Ladang bunga.jpg



F. METHOD MOVE

Jika karena sesuatu hal kita tidak bisa (atau tidak ingin) menggunakan symlink, Laravel juga menyediakan cara agar file yang di upload langsung tersimpan ke folder public, tidak harus lewat folder storage. Caranya, gunakan method move():

1. Modifikasi controller FileUploadController.php

Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-12 : FILE UPLOAD

Mata Kuliah Praktikum Pemrograman Web Lanjut (PWL)

Mei 2024

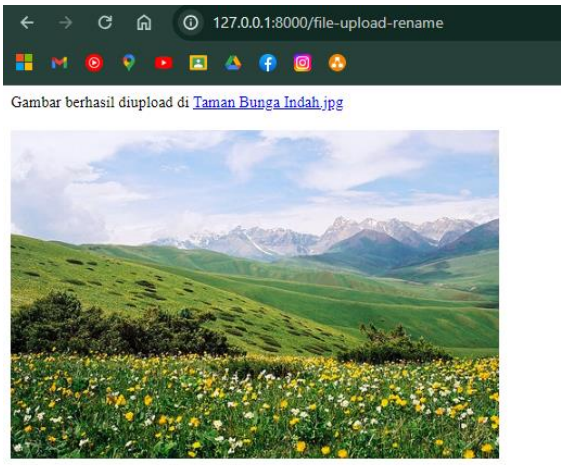
```
$request->validate([
    'berkas'=>'required|file|image|max:500',]);
$textFile = $request->berkas->getClientOriginalName();
$namaFile ='web'.time()."_".$textFile;

$path = $request->berkas->move('gambar', $namaFile);
$path = str_replace("\\","//", $path);
echo "variable path berisi:$path <br>";

$pathBaru=asset('gambar/'.$namaFile);
echo "proses upload berhasil, data disimpan pada:$path";
echo "<br>";
echo "Tampilkan link: <a href='$pathBaru'>$pathBaru</a>";
```

variable path berisi:gambar//web1716971880.Ladang bunga.jpg
proses upload berhasil, data disimpan pada:gambar//web1716971880.Ladang bunga.jpg
Tampilkan link: http://127.0.0.1:8000/gambar/web1716971880.Ladang bunga.jpg

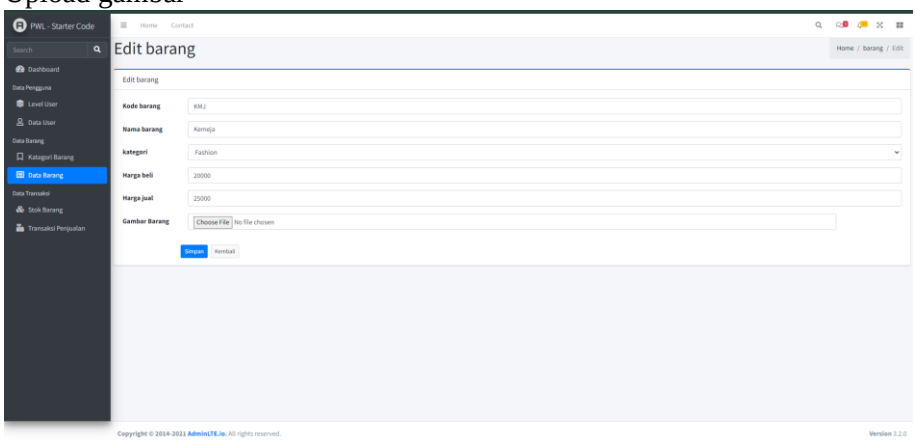
Di baris 22 menggunakan perintah $request->berkas->move('image',$namaFile) untuk memindahkan file upload. Sama seperti method store() dan storeAs(), method move() butuh 2 argument. Argument pertama diisi dengan nama folder penyimpanan, dan argument kedua berupa nama file.
Hasilnya, di folder public akan muncul folder image yang berisi file http://127.0.0.1:8000/gambar/web-1714803545.images.png. Karena sudah berada di dalam folder public, maka file ini bisa langsung di akses dari web browser. Tambahan fungsi str_replace() di baris 23 bertujuan untuk mengganti tanda pemisah folder dari forward slash "\" menjadi back slash "/" untuk variabel $path. Misalnya dari gambar\web1643167529.jpg menjadi gambar/web1643167529.jpg. Tanda pemisah folder ini sebenarnya tergantung jenis OS yang dipakai. Di OS Windows, alamat http://localhost:8000/gambar\web-1643167529.jpg tetap bisa diakses. Namun akan lebih rapi jika semua pemisah folder menggunakan karakter back slash "/".

TUGAS

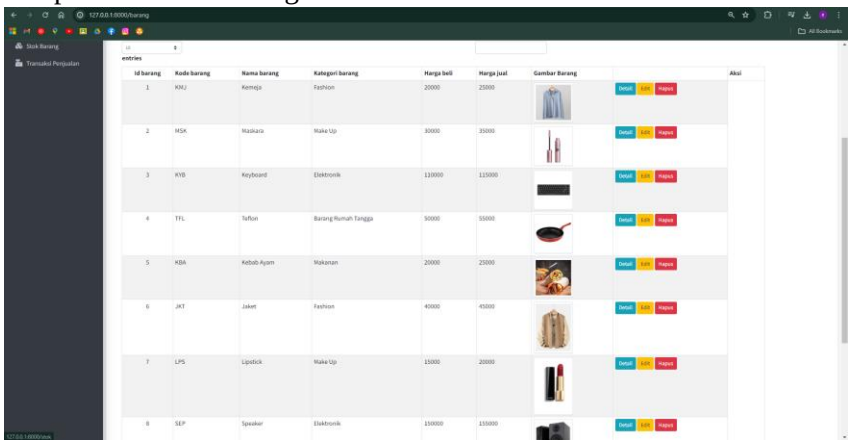
1. buat form file upload dimana kita bisa menentukan sendiri nama file akhir. Nama file ini diambil dari inputan text box yang juga ada di dalam form tersebut. Setelah di submit, langsung tampilkan file tersebut di web browser menggunakan tag <img>.
   Jawaban:



2. Tambahkan proses upload image dan application pada starter code
   Jawaban:
   A. Barang
   a. Upload gambar

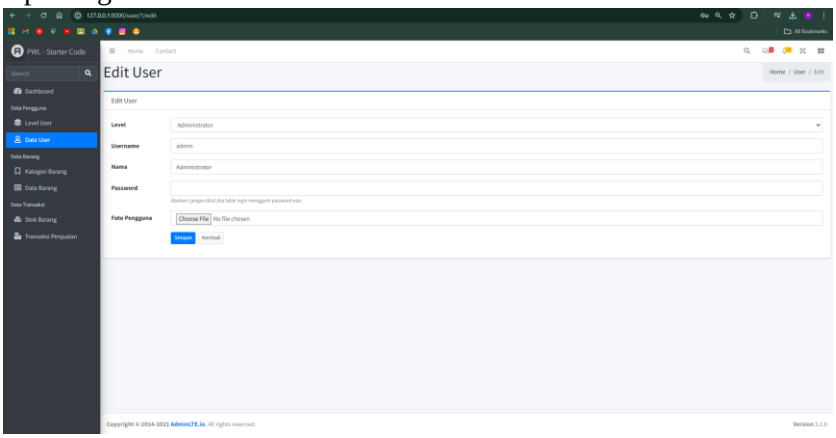


b. Tampilan setelah diberi gambar



B. User

a. Upload gambar



b. Hasil setelah diberi gambar