

LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT



DISUSUN OLEH:
THORIQ FATHURROZI
2H/28
2241720052

PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2024



Nama	:	Thoriq Fathurrozi
Nim	:	2241720052
Kelas	:	2H

A. PERSIAPAN AWAL	
1.	<p>Membuat beberapa file yang diperlukan, yang pertama controller :</p> <p>php artisan make:controller FileUploadController</p> <ul style="list-style-type: none">● frrzy@Thoriqs-MacBook-Pro PWL_POS % php artisan make:controller FileUploadControllerINFO Controller [app/Http/Controllers/FileUploadController.php] created successfully.○ frrzy@Thoriqs-MacBook-Pro PWL_POS %
2.	<p>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.</p> <pre>http://Controllers>FileUploadController.php>Intelephense></pre> <pre><?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) { return "Pemerolehan file upload disini"; } }</pre>



Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

3.

Membuat route pada web.php

```
5 | use App\Http\Controllers\FileUploadController;      You, 1 second ago • Uncommitted
6 |
7 |
8 |/*
9 | -----
10| / Web Routes
11| -----
12|
13| / Here is where you can register web routes for your application. These
14| / routes are loaded by the RouteServiceProvider and all of them will
15| / be assigned to the "web" middleware group. Make something great!
16|
17| */
18|
19| Route::get('/', [WelcomeController::class, 'index']);
20|
21|
22| Route::get('/file-upload', [FileUploadController::class, 'fileUpload']);
23| Route::post('/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-QWTKZyjpPEjISv5WaRU9FeRpk6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" 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" required>
                @error('berkas')
                    <div class="text-danger">{{ $message }}</div>
                @enderror
            </div>
            <button type="submit" class="btn btn-primary my-2">Upload</button>
        </form>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-YVpcryf0tY3lHB60NNkmXc5s9fDV2LEsaAA55NDz0xhy9GkcidslK1eN7N6jIeHz" crossorigin="anonymous">
    </script>
</body>

</html>
```



Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

Gunakan CDN Bootstrap : <https://getbootstrap.com/> agar tampilan lebih menarik
Sehingga tampilan menjadi seperti berikut :

A screenshot of a web browser window showing a file upload interface. The URL in the address bar is `localhost:8000/file-upload`. The page title is "File Upload". Below the title is a label "Gambar Profile". There is a file input field with the placeholder "No file selected.". Below the input field is a blue "Upload" button.

Untuk membuat tampilan form, menggunakan sedikit class CSS bawaan Bootstrap (Gunakan CDN Bootstrap). Karena kita akan mengupload file, maka di dalam tag <form> 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, <input type="file" name="berkas">. 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":

A screenshot of a web browser window showing a file upload interface. The URL in the address bar is `localhost:8000/file-upload`. The page title is "File Upload". Below the title is a label "Gambar Profile". There is a file input field showing the file "Buletin - News Website.jpg" selected. Below the input field is a blue "Upload" button.

A screenshot of a web browser window showing a file upload interface. The URL in the address bar is `localhost:8000/file-upload`. The page title is "File Upload". Below the title is a label "Gambar Profile". There is a file input field. Below the input field is the text "Pemerosesan file upload disini".



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**:

```
public function prosesFileUpload(Request $request)
{
    dump($request->berkas);      You, 1 second ago • Uncollected
    // dump($request->file('file'));
    // return "Pemrosesan file upload disini";
```

Isi dari method ini hanya 1 baris, langsung **men-dump \$request->berkas**. Nama "berkas" ini berasal dari nilai atribut name pada tag `<input type="file" class="form-control" id="berkas" name="berkas">`.

```
<input type="file" class="form-control" id="berkas" name="berkas">
```

2. lakukan 2 percobaan, **pertama langsung submit form tanpa mengisi file apapun. Dankedua, upload file sembarang dan submit.** Screen Shot bagaimana hasilnya?
tidak ada berkas yang dipilih

File Upload

Gambar Profile

No file selected.

```
null // app/Http/Controllers/FileUploadController.php:16
```

terdapat berkas yang diupload

File Upload

Gambar Profile

Finsy - Finance Landing Page 📸.jpg



Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

	<pre>Illuminate\Http\UploadedFile {#1567 ▼ // app/Http/Controllers/FileUploadController.php:16 -test: false -originalName: "Finsy - Finance Landing Page 📸.jpg" -mimeType: "image/jpeg" -error: 0 #hashName: null path: "/private/var/folders/cv/b5nsdb9d2lxgkt3ndtsj3qq80000gn/T" filename: "phpxTwfiW" basename: "phpxTwfiW" pathname: "/private/var/folders/cv/b5nsdb9d2lxgkt3ndtsj3qq80000gn/T/phpxTwfiW" extension: "" realPath: "/privat.../T/phpxTwfiW" aTime: 2024-05-28 04:30:39 mTime: 2024-05-28 04:30:39 cTime: 2024-05-28 04:30:39 inode: 8844771 size: 49082 perms: 0100600 owner: 501 group: 20 type: "file" writable: true readable: true executable: false file: true dir: false link: false }</pre> <ul style="list-style-type: none">• Jika tidak ada file yang di upload, perintah <code>\$request->berkas</code> 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 <code>\$request->hasFile('berkas')</code>. 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) <pre>public function prosesFileUpload(Request \$request) { // dump(\$request->berkas); // dump(\$request->file('file')); // return "Pemerosesan file upload disini"; if (\$request->hasFile('berkas')) { echo "path(): " . \$request->berkas->path() . "
"; echo "extension(): " . \$request->berkas->extension() . "
"; You, 4 seconds ago · Uncommitt... echo "getClientOriginalExtension(): " . \$request->berkas->getClientOriginalExtension() . "
"; echo "getMimeType(): " . \$request->berkas->getMimeType() . "
"; echo "getClientOriginalName(): " . \$request->berkas->getClientOriginalName() . "
"; echo "getSize(): " . \$request->berkas->getSize() . "
"; } else { return "Tidak ada berkas yang diupload"; } }</pre>
4.	Lakukan percobaan upload file sehingga akan muncul keterangan sbb:



File Upload	
Gambar Profile	<input type="file"/> Finsy - Finance Landing Page 📸.jpg
	<input type="button" value="Upload"/>
Penjelasan :	<p>Di baris 18 pada method prosesFileUpload terdapat sebuah kondisi if yang akan dijalankan jika \$request->hasFile('berkas') bernilai true. Di dalam blok ini, untuk mengakses beberapa method. File sample yang diupload adalah sebuah file excel . Berikut penjelasan dari method prosesFileUpload yang ada di baris 20 - 32:</p> <ul style="list-style-type: none">• \$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-DIV-2C_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() . "lulus validasi";
```

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

File Upload

Gambar Profile

No file selected.

The berkas field is required.

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() . "lulus 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**.

```
[frrzy@Thoriqs-MacBook-Pro etc % pwd
/Applications/XAMPP/etc
[frrzy@Thoriqs-MacBook-Pro etc % cat php.ini | grep max_file
upload_max_filesize=40M
max_file_uploads=20
frrzy@Thoriqs-MacBook-Pro etc %
```



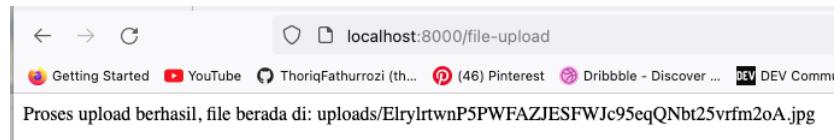
Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

D. MEMINDAH FILE UPLOAD

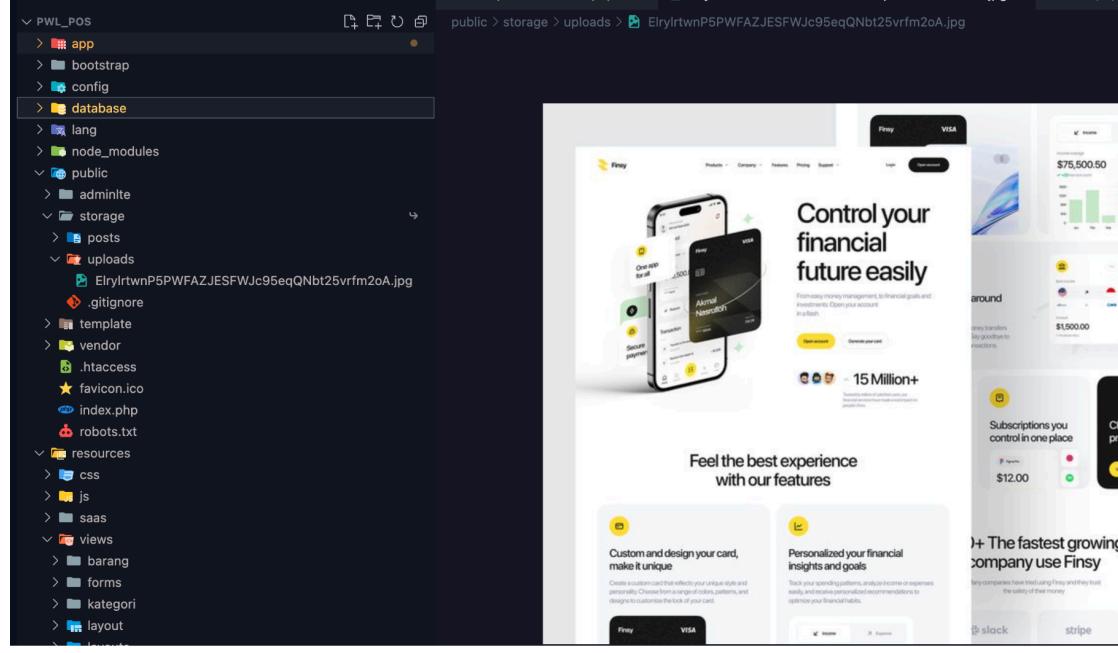
1. Cara pertama adalah menggunakan method store() yang bisa diakses dari Request object. Berikut contoh penggunaannya:

```
$request->validate([
    'berkas' => 'required|file|image|max:500',
]);
$path = $request->berkas->store('uploads');
return "Proses upload berhasil, file berada di: $path"; // echo $request->berkas->getClientOriginalName() . "lalu validasi";
```

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:



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





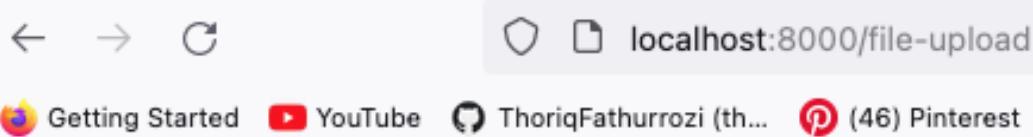
file yang di upload sudah bisa diakses. Namun kenapa nama file acak seperti itu? Method store() memang akan **men-generate nama acak untuk setiap file yang di upload**. Kesannya memang agak aneh, tapi sebenarnya sangat bermanfaat:

- Menghindari kemungkinan file ditimpas. 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.

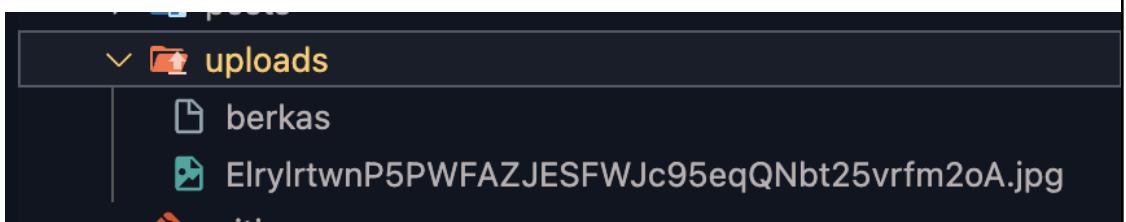
Bagaimana jika kita tetap ingin membuat nama file sendiri? Tidak masalah, Laravel menyediakan method storeAs() untuk keperluan ini. Berikut contoh penggunaannya: (line 20)

```
$request→validate([
    'berkas' => 'required|file|image|max:500',
]);
// $path = $request→berkas→store('uploads');
$path = $request→berkas→storeAs('uploads', 'berkas'); You, 1 sec
return "Proses upload berhasil, file berada di: $path";
// echo $request→berkas→getClientOriginalName() . "lulus validasi";
```

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)



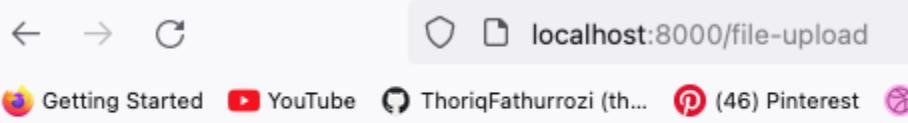
```
$request->validate([
    'berkas' => 'required|file|image|max:500',
]);
$namaFile = $request->berkas->getClientOriginalName();
$path = $request->berkas->storeAs('uploads', $namaFile);
return "Proses upload berhasil, data disimpan pada: $path";
```

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:

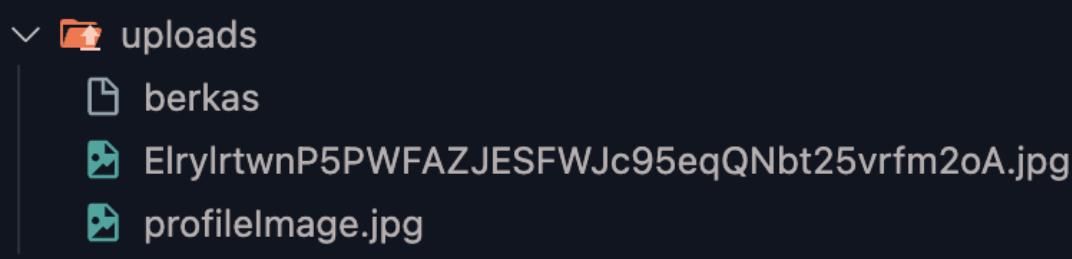
File Upload

Gambar Profile

profileImage.jpg



Proses upload berhasil, data disimpan pada: uploads/profileImage.jpg



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](#))

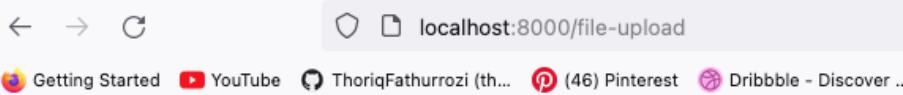
```
$request->validate([
    'berkas' => 'required|file|image|max:500',
]);
$extFile = $request->berkas->getClientOriginalName();
$namaFile = 'web-' . time() . "." . $extFile;
$path = $request->berkas->storeAs('uploads', $namaFile);
return "Proses upload berhasil, data disimpan pada: $path";
```

Di baris 19, 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`.

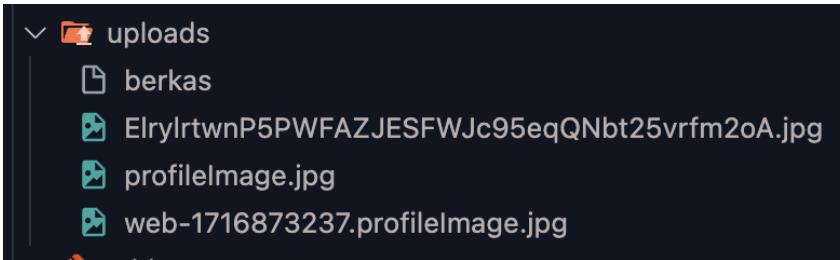
File Upload

Gambar Profile

profileImage.jpg



Proses upload berhasil, data disimpan pada: uploads/web-1716873237.profileImage.jpg



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.

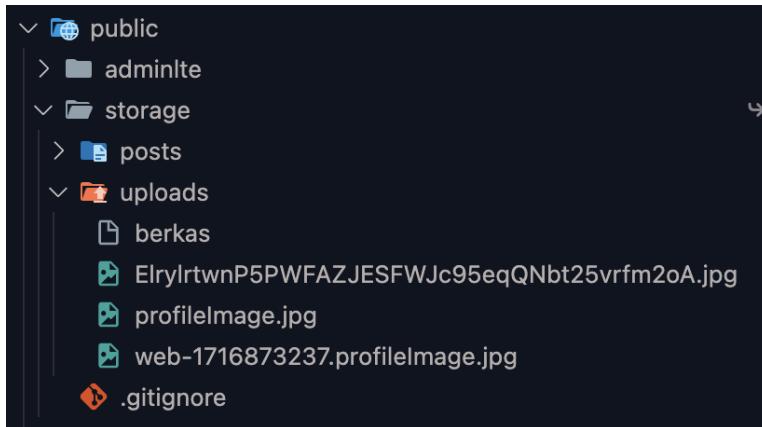


E. MEMBUAT SYMLINK

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

```
frrzy@Thoriqs-MacBook-Pro PWL_POS % php artisan storage:link
INFO  The [public/storage] link has been connected to [storage/app/public].
```

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



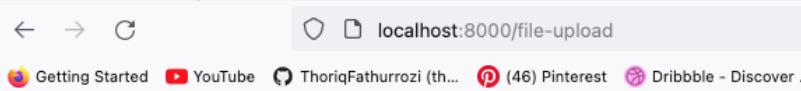
Ketika di klik, terdapat file .gitignore di dalam symlink storage. Ini hanya file bantu yang berfungsi sebagai tanda agar isi folder storage tidak perlu di backup oleh aplikasi Git. Symlink storage pada dasarnya merupakan shortcut atau mirror dari folder storage\app\public.

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 dihapus, 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():

```
$namaFile = 'web-' . time() . "." . $extFile;
$path = $request->berkas->storeAs('public', $namaFile);
```

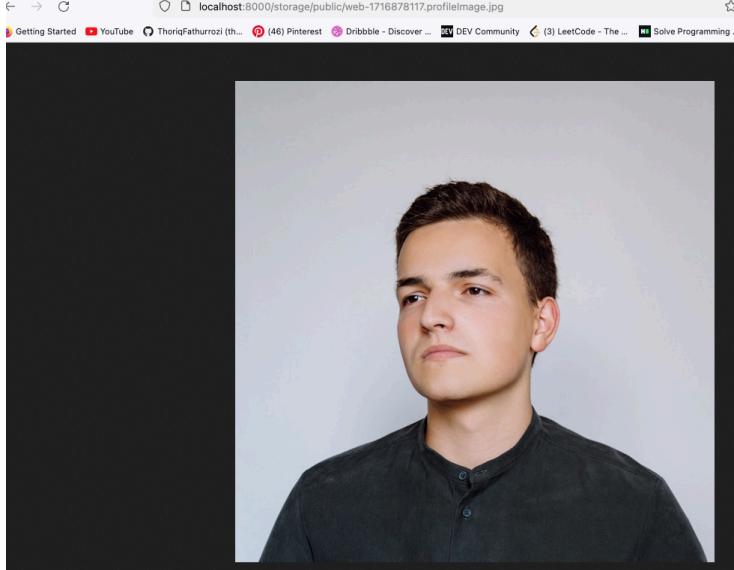
Perhatikan bahwa argumen pertama dari method storeAs() adalah 'public'(line 21). Inilah folder tempat file upload akan disimpan. Silahkan upload sebuah file, maka akan tampil hasil berikut:



Proses upload berhasil, data disimpan pada: public/web-1716878117.profileImage.jpg



Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

	<p>Proses upload berhasil, dan file tersebut ada di storage\app\public\ web-1714801950.images.png. Nama file yang akan anda dapatkan 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/ web-1714801950.images.png</p> 
3.	<p>Menambahkan link agar gambar dapat diakses, tambahkan kode pada method prosesFileUpload:</p> <pre>\$request->validate(['berkas' => 'required file image max:500',]); \$extFile = \$request->berkas->getClientOriginalName(); \$namaFile = 'web-' . time() . "." . \$extFile; \$path = \$request->berkas->storeAs('public', \$namaFile); \$pathBaru = asset('storage/public/' . \$namaFile); echo "Proses upload berhasil, data disimpan pada: \$path
"; return "Tampilkan link: \$pathBaru";</pre>



Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

File Upload

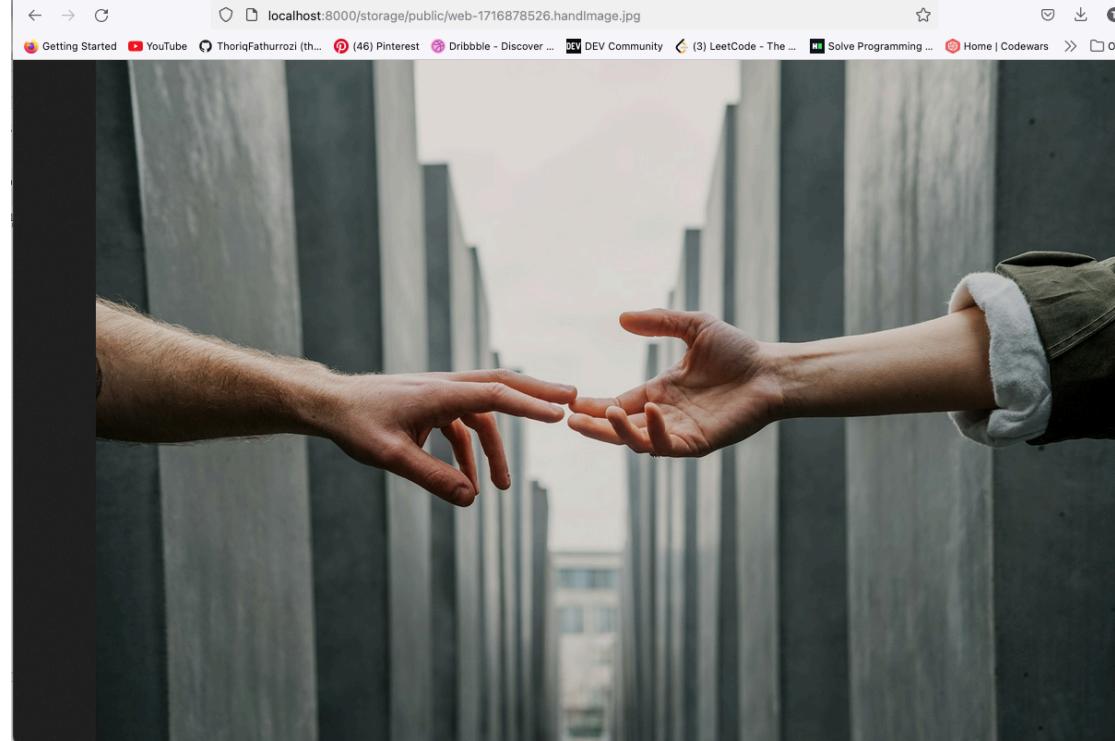
Gambar Profile

handImage.jpg

localhost:8000/file-upload

Getting Started YouTube ThoriqFathurrozi (th... Pinterest Dribbble - Discov

Proses upload berhasil, data disimpan pada: public/web-1716878526.handImage.jpg
Tampilkan link: <http://localhost:8000/storage/public/web-1716878526.handImage.jpg>





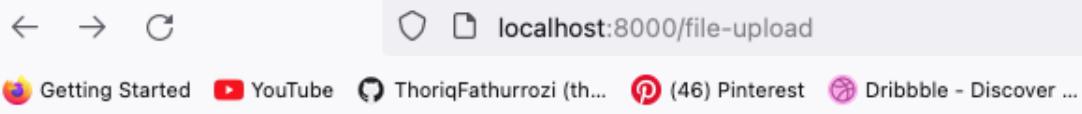
F. METHOD MOVE

1. Modifikasi controller FileUploadController.php

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

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

$pathBaru = asset('gambar/' . $namaFile);      You, 1 second ago •
echo "Proses upload berhasil, data disimpan pada: $path <br>";
return "Tampilkan link: <a href='$pathBaru'$>$pathBaru</a>";
```



Variabel path berisi: gambar/web-1716879729.handImage.jpg
Proses upload berhasil, data disimpan pada: gambar/web-1716879729.handImage.jpg
Tampilkan link: <http://localhost:8000/gambar/web-1716879729.handImage.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 diakses 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

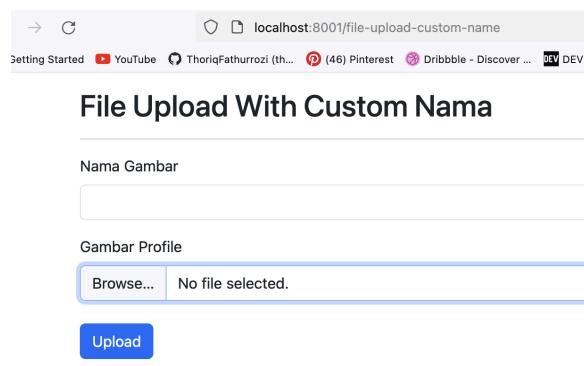
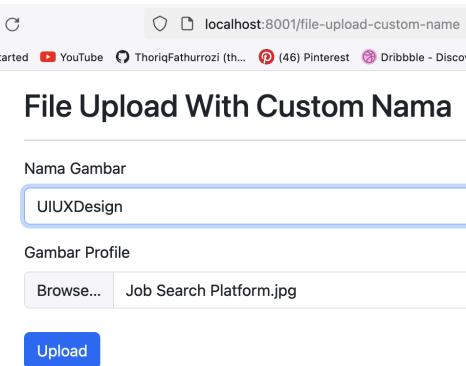
gambar\web1643167529.jpg menjadi gambar\web1643167529.jpg. Tanda pemisah folder ini sebenarnya tergantung jenis OS yang dipakai. Di OS Windows, alamat <http://localhost:8000/gambar\web-1643167529.jpg> tetap bisa diakses. Namun akan lebih rapi jika semua pemisah folder menggunakan karakter back slash "/".



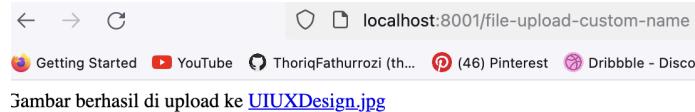
Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

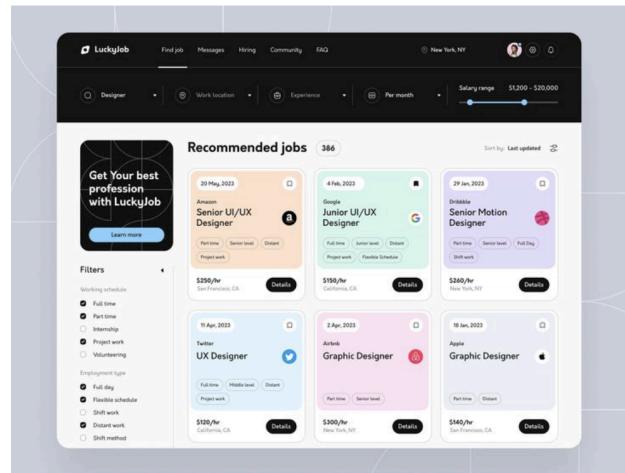
Tugas

1.

	
---	--

ketika berhasil di upload





2.

web route

```
Route::get('/file-upload-custom-name', [FileUploadController::class, 'fileUploadCustomName']);  
Route::post('/file-upload-custom-name', [FileUploadController::class, 'prosesFileUploadCustomName']);
```



Jurusan Teknologi Informasi Politeknik Negeri Malang
Pemrograman Web Lanjut
Mata Kuliah Pemrograman Web Lanjut
Pengampu: Tim Ajar Pemrograman Web Lanjut
May 2024

Controller fileUpload with function custom name

```
1 reference | 0 overrides
public function fileUploadCustomNama()
{
    return view('file-upload-custom-nama');
}

1 reference | 0 overrides
public function prosesFileUploadCustomNama()
{
    request()→validate([
        'namaBerkas' => 'required|max:255',
        'berkas' => 'required|file|image|max:500'
    ]);

    $file = request()→file('berkas');

    $ext = $file→getClientOriginalExtension();
    $newName = request()→namaBerkas . "." . $ext;      You, 1 second ago · Unc
    $path = $file→storeAs('uploads', $newName);

    echo "Gambar berhasil di upload ke <a href='storage/$path'>$newName</a>";
    echo "<br><br>";
    echo "<img width=500 src='storage/$path' />";
}
```

view file-upload-custom-nama

```
<!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-QWTKZyjpEPISv5WaNkXm0vGQVf2J7PdIYc0E4xWl9D0QF1q+uJG4&lt;!--> crossorigin="anonymous">
    <title>File Upload Custom nama</title>
</head>

<body>
    <div class="container mt-3">
        <h2>File Upload With Custom Name</h2>
        <hr>
        <form action="{{ url('/file-upload-custom-name') }}" method="POST" enctype="multipart/form-data">
            @csrf
            <div class="mb-3">
                <label for="namaBerkas" class="form-label">Nama Gambar</label>
                <input type="text" class="form-control" id="namaBerkas" name="namaBerkas">
                @error('namaBerkas')
                    <div class="text-danger">{{ $message }}</div>
                @enderror
            </div>
            <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>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-YvpcrYf0tY3lHB69NNkmXc5s9fDVZLESaAA55NDz0xhy96kcIdslK1eN7N6jIeHz" crossorigin="anonymous">
    </script>
</body>
```