



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

JOBSHEET 12

FILE UPLOAD

Upload file menjadi salah fitur yang sering kita jumpai. Misalnya untuk form registrasi, tidak jarang perlu mengupload foto profil, upload berkas pdf, dsb. Laravel sudah menyediakan fitur untuk memproses file upload, termasuk mengelola lokasi file.

Sesuai dengan **studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. PERSIAPAN AWAL

1. Membuat beberapa file yang diperlukan, yang pertama controller :

php artisan make:controller FileUploadController

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\laragon\www\PWL2024\Materi\jobsheet12> php artisan make:controller FileUploadController
INFO Controller [C:\laragon\www\PWL2024\Materi\jobsheet12\app\Http\Controllers\FileUploadController.php] created successfully.
PS C:\laragon\www\PWL2024\Materi\jobsheet12> █
```

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**.



```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class FileUploadController extends Controller
8  {
9      //
10     public function fileUpload(){
11         return view('file-upload');
12     }
13
14     public function prosesFileUpload(Request $request){
15         return "Pemrosesan file upload di sini";
16     }
17 }
18
```

3. Membuat route pada web.php

```
file-upload.blade.php ...\01. Persiapan Awal\...  file-upload.blade.php ...resources\...  web.php X
Materi > jobsheet12 > routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\FileUploadController;
5
6
7  /*
8  |-----
9  | Web Routes
10 |-----
11 |
12 | Here is where you can register web routes for your application. These
13 | routes are loaded by the RouteServiceProvider and all of them will
14 | be assigned to the "web" middleware group. Make something great!
15 |
16 */
17
18 Route::get('/', function () {
19     return view('welcome');
20 });
21 Route::get('/file-upload', [FileUploadController::class, 'fileUpload']);
22 Route::post('/file-upload', [FileUploadController::class, 'prosesfileUpload']);
23
```

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 menajdi tujuan saat form di submit.

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

```
file-upload.blade.php X
Materi > jobsheet12 > resources > views > file-upload.blade.php > html > body > div.container.mt-3 > hr
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   {{!-- <link href="/css/bootstrap.min.css" rel="stylesheet" --}}
9   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
10     integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJy66hW+ALEwIH" crossorigin="anonymous">
11   <title>File Upload</title>
12 </head>
13
14 <body>
15
16   <div class="container mt-3">
17     <h2>File Upload</h2>
18     <hr>
19
20     <form action="{{ url('/file-upload') }}" method="POST" enctype="multipart/form-data">
21       @csrf
22
23       <div class="mb-3">
24         <label for="berkas" class="form-label">Gambar Profile</label>
25         <input type="file" class="form-control" id="berkas" name="berkas">
26         @error('berkas')
27           <div class="text-danger">{{ $message }}</div>
28         @enderror
29       </div>
30
31       <button type="submit" class="btn btn-primary my-2">Upload</button>
32     </form>
33   </div>
34
35 </body>
36
37 </html>
38
```

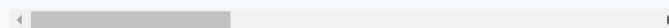
Gunakan CDN Bootstrap : <https://getbootstrap.com/> agar tampilan lebih menarik



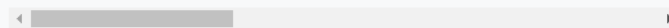
Include via CDN

When you only need to include Bootstrap's compiled CSS or JS, you can use [jsDelivr](#). See it in action with our simple [quick start](#), or [browse the examples](#) to jumpstart your next project. You can also choose to include Popper and our JS [separately](#).

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/b
```



```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/b
```





Sehingga tampilan menjadi seperti berikut :

File Upload

Gambar Profile

Choose File No file chosen

Upload

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":

File Upload

Gambar Profile

Choose File FormatNilai-DIV-2C_SIB - 2C-Pemrograman_Web.xls

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`**:



```
FileUploadController.php X file-upload.blade.php
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  24 references | 0 implementations
8  class FileUploadController extends Controller
9  {
10     //
11     7 references | 0 overrides
12     public function fileUpload(){
13         return view('file-upload');
14     }
15
16     7 references | 0 overrides
17     public function prosesFileUpload(Request $request){
18         dump($request->berkas);
19         // dump($request->file('file'));
20         // return "Pemrosesan file upload di sini";
21     }
22 }
```

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" name="berkas">`.

```
FileUploadController.php file-upload.blade.php X
Materi > jobsheet12 > resources > views > file-upload.blade.php > html > body > div.container.mt-3 > form > div.mb-3 > input#berkas.form-control
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      {{-- <link href="/css/bootstrap.min.css" rel="stylesheet" -->}}
9      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
10         integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhY66HwA&LEWlH" crossorigin="anonymous">
11      <title>File Upload</title>
12  </head>
13
14  <body>
15
16      <div class="container mt-3">
17          <h2>File Upload</h2>
18          <hr>
19
20          <form action="{{ url('/file-upload') }}" method="POST" enctype="multipart/form-data">
21              @csrf
22
23              <div class="mb-3">
24                  <label for="berkas" class="form-label">Gambar Profile</label>
25                  <input type="file" class="form-control" id="berkas" name="berkas">
26                  @error('berkas')
27                      <div class="text-danger">{{ $message }}</div>
28                  @enderror
29              </div>
30
31              <button type="submit" class="btn btn-primary my-2">Upload</button>
32          </form>
33      </div>
34
35  </body>
36
37  </html>
```



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

- 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)

```
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\Http\Controllers\FileUploadController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  24 references | 0 implementations
8  class FileUploadController extends Controller
9  {
10     //
11     7 references | 0 overrides
12     public function fileUpload(){
13         return view('file-upload');
14     }
15
16     7 references | 0 overrides
17     public function prosesFileUpload(Request $request){
18         // dump($request->berkas);
19         // dump($request->file('file'));
20         // return "Pemrosesan file upload di sini";
21         if($request->hasFile('berkas'))
22         {
23             echo "path(): ".$request->berkas->path();
24             echo "<br>";
25             echo "extension(): ".$request->berkas->extension();
26             echo "<br>";
27             echo "getClientOriginalExtension(): ".$request->berkas->getClientOriginalExtension();
28             echo "<br>";
29             echo "getMimeType(): ".$request->berkas->getMimeType();
30             echo "<br>";
31             echo "getClientOriginalName(): ".$request->berkas->getClientOriginalName();
32             echo "<br>";
33             echo "getSize(): ".$request->berkas->getSize();
34         }
35         else
36         {
37             echo "Tidak ada berkas yang diupload";
38         }
39     }
40 }
```



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

```
← ↻ ⓘ 127.0.0.1:8000/file-upload  
path(): C:\Users\Dimas_Polinema\AppData\Local\Temp\php5674.tmp  
extension(): xls  
getClientOriginalExtension(): xls  
getMimeType(): application/vnd.ms-excel  
getClientOriginalName(): FormatNilai-DIV-2C_SIB - 2C-Pemrograman_Web.xls  
getSize(): 845312
```

Penjelasan :

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:

- **`$request->berkas->path()`**, menampilkan alamat path dari file yang sudah di upload. Perhatikan bahwa isinya adalah alamat temporary dari pengaturan PHP. Dalam contoh ini, alamat file yang diupload akan disimpan di `C:\Users\Dimas_Polinema\AppData\Local\Temp\php5674.tmp`.
- **`$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

File yang di upload tentunya butuh proses validasi. Malah ini menjadi lebih penting karena tentu kita tidak ingin ada yang mengupload file berbahaya, atau malah mengupload file .php yang berisi kode/script tertentu.

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`)

```
FileUploadController.php
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  24 references | 0 implementations
class FileUploadController extends Controller
8  {
9      //
10     7 references | 0 overrides
    public function fileUpload(){
11         return view('file-upload');
12     }
13
14     7 references | 0 overrides
    public function prosesFileUpload(Request $request){
15         // dump($request->berkas);
16         // dump($request->file('file'));
17         // return "Pemrosesan file upload di sini";
18         $request->validate([
19             'berkas'=>'required',]);
20         echo $request->berkas->getClientOriginalName()."lolos validasi";
21     }
22 }
23
24 }
25
26
```

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



← ↻ ⓘ 127.0.0.1:8000/file-upload

File Upload

Gambar Profile

Choose File

No file chosen

The berkas field is required.

Upload

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)

```
FileUploadController.php •
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > ...
7  class FileUploadController extends Controller
10     public function fileUpload(){
12
13     }
14
15     7 references | 0 overrides
16     public function prosesFileUpload(Request $request){
17         // dump($request->berkas);
18         // dump($request->file('file'));
19         // return "Pemrosesan file upload di sini";
20         $request->validate([
21             'berkas'=>'required|file|image|max:500',]);
22         echo $request->berkas->getClientOriginalName()."lolos validasi";
23     }
24 }
25
```

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**.

```
826 ; https://php.net/cgi.rfc2616-headers
827 ;cgi.rfc2616_headers = 0
828
829 ; cgi.check_shebang_line controls whether CGI PHP checks for line starting with #!
830 ; (shebang) at the top of the running script. This line might be needed if the
831 ; script support running both as stand-alone script and via PHP CGI<. PHP in CGI
832 ; mode skips this line and ignores its content if this directive is turned on.
833 ; https://php.net/cgi.check-shebang-line
834 ;cgi.check_shebang_line=1
835
836 ;;;;;;;;;;;;;;;;;
837 ; File Uploads ;
838 ;;;;;;;;;;;;;;;;;
839
840 ; Whether to allow HTTP file uploads.
841 ; https://php.net/file-uploads
842 file_uploads = On
843
844 ; Temporary directory for HTTP uploaded files (will use system default if not
845 ; specified).
846 ; https://php.net/upload-tmp-dir
847 ;upload_tmp_dir =
848
849 ; Maximum allowed size for uploaded files.
850 ; https://php.net/upload-max-filesize
851 upload_max_filesize = 2G
852
853 ; Maximum number of files that can be uploaded via a single request
854 max_file_uploads = 20
---
```

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:



```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\Http\Controllers\FileUploadController > prosesFileUpload()
7 class FileUploadController extends Controller
10 public function fileUpload(){
12
13 }
14
15 7 references | 0 overrides
16 public function prosesFileUpload(Request $request){
17
18     $request->validate([
19         'berkas' => 'required|file|image|max:500',]);
20     $path = $request->berkas->store('uploads');
21     echo "proses upload berhasil, file berada di: $path";
22     // echo $request->berkas->getClientOriginalName().".lolos validasi";
23
24
25
26
27 }
```

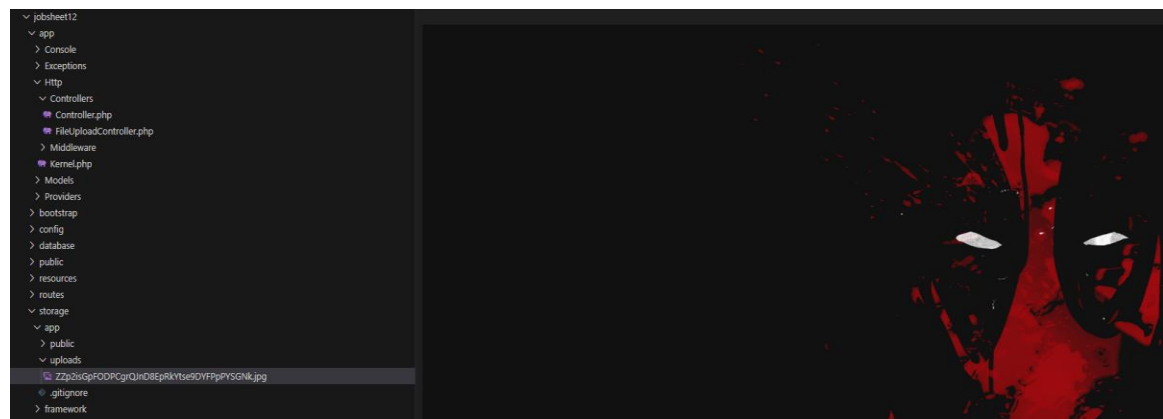
Di baris 17-20 terdapat kode untuk proses validasi yang sudah di bahas sebelumnya. Proses pemindahan file dilakukan di baris **19** dengan perintah **\$request->berkas->store('uploads')**. Method **store()** akan mengambil file upload dari folder temporary dan memindahkannya ke folder **storage/app** 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:

← ↻ ⓘ 127.0.0.1:8000/file-upload

proses upload berhasil, file berada di: uploads/ZZp2isGpFODPCgrQJnD8EpRkYtse9DYFPpPYSGNk.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**.

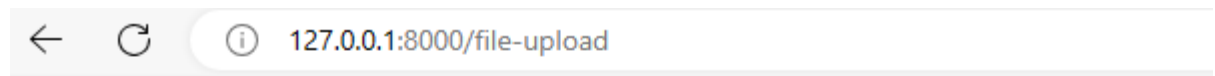
Kesannya memang agak aneh, tapi sebenarnya sangat bermanfaat:

- 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.

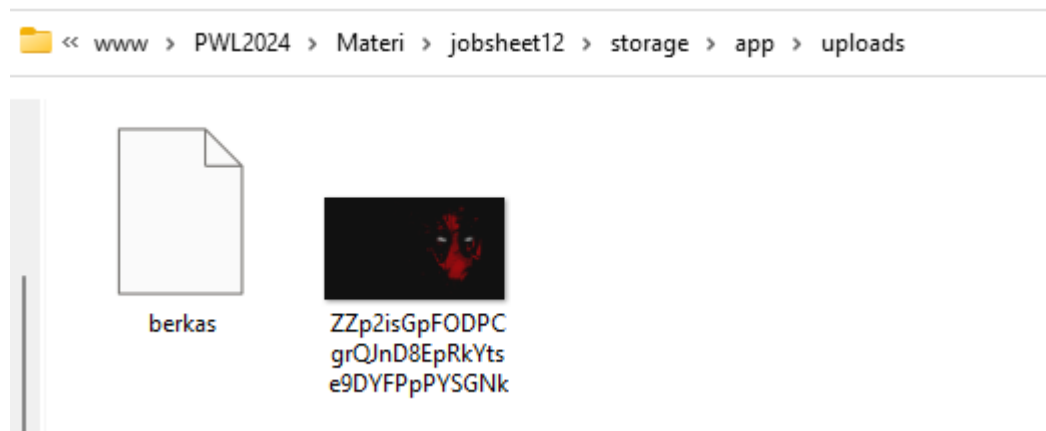
Bagaimana jika kita tetap ingin membuat nama file sendiri? Tidak masalah, Laravel menyediakan method `storeAs()` untuk keperluan ini. Berikut contoh penggunaannya: (line 20)

```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > ...
7  class FileUploadController extends Controller
10     public function fileUpload(){
12
13     }
14
15     7 references | 0 overrides
16     public function prosesFileUpload(Request $request){
17
18         $request->validate([
19             'berkas'=>'required|file|image|max:500',]);
20         // $path = $request->berkas->store('uploads');
21         $path = $request->berkas->storeAs('uploads','berkas');
22         echo "proses upload berhasil, file berada di: $path";
23         // echo $request->berkas->getClientOriginalName()."lolos validasi";
24     }
25 }
26
27
```

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".

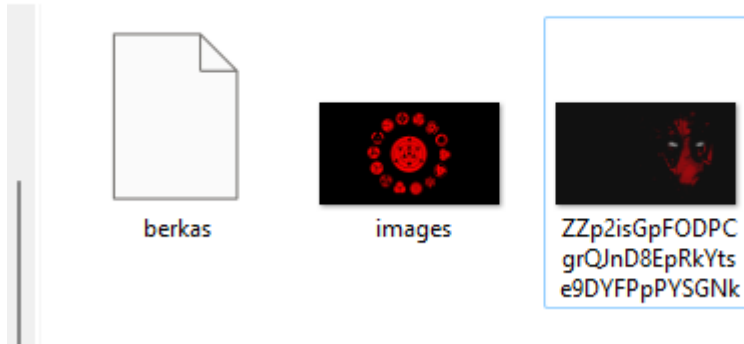
2. mengambil nama file dari fungsi **getClientOriginalName()** seperti contoh berikut:
(app/Http/Controllers/FileUploadController.php)

```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\<Http\>Controllers\FileUploadController > prosesFileUpload()
7  class FileUploadController extends Controller
10  public function fileUpload(){
12
13  }
14
15  7 references | 0 overrides
16  public function prosesFileUpload(Request $request){
17
18      $request->validate([
19          'berkas'=>'required|file|image|max:500',]);
20      $namaFile=$request->berkas->getClientOriginalName();
21      $path = $request->berkas->storeAs('uploads',$namaFile);
22      echo "proses upload berhasil, data disimpan pada:$path";
23  }
24  }
```

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:

« www > PWL2024 > Materi > jobsheet12 > storage > app > uploads



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**)

```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\Http\Controllers\FileUploadController > prosesFileUpload()
7 class FileUploadController extends Controller
10 public function fileUpload(){
12
13 }
14
15 7 references | 0 overrides
16 public function prosesFileUpload(Request $request){
17
18     $request->validate([
19         'berkas' => 'required|file|image|max:500',]);
20     $extfile=$request->berkas->getClientOriginalName();
21     $namaFile='web-'.time().".".$extfile;
22     $path = $request->berkas->storeAs('uploads',$namaFile);
23     echo "proses upload berhasil, data disimpan pada:$path";
24
25 }
26
27
```

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`.



« www » PWL2024 » Materi » jobsheet12 » storage » app » uploads



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 (mirroring). 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\PWL2024\Materi\jobsheet12> php artisan storage:link
[INFO] The [C:\laragon\www\PWL2024\Materi\jobsheet12\public\storage] link has been connected to [C:\laragon\www\PWL2024\Materi\jobsheet12\storage\app\public].
PS C:\laragon\www\PWL2024\Materi\jobsheet12>
```

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



<< OS (C:) > laragon > www > PWL2024 > Materi > jobsheet12 > public

▼ ↺

Name	Date modified	Type	Size
 storage	5/4/2024 12:38 PM	File folder	
 .htaccess	2/13/2024 9:23 AM	HTACCESS File	1 KB
 favicon	2/13/2024 9:23 AM	Icon	0 KB
 index	2/13/2024 9:23 AM	PHP Source File	2 KB
 robots	2/13/2024 9:23 AM	Text Document	1 KB

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():

```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\Http\Controllers\FileUploadController > prosesFileUpload()
7  class FileUploadController extends Controller
10  public function fileUpload(){
12
13  }
14
15  7 references | 0 overrides
16  public function prosesFileUpload(Request $request){
17
18      $request->validate([
19          'berkas' => 'required|file|image|max:500', ]);
20      $extfile=$request->berkas->getClientOriginalName();
21      $namaFile='web-'.time().".".$extfile;
22      $path = $request->berkas->storeAs('public',$namaFile);
23      echo "proses upload berhasil, data disimpan pada:$path";
24  }
25  }
26
27
```

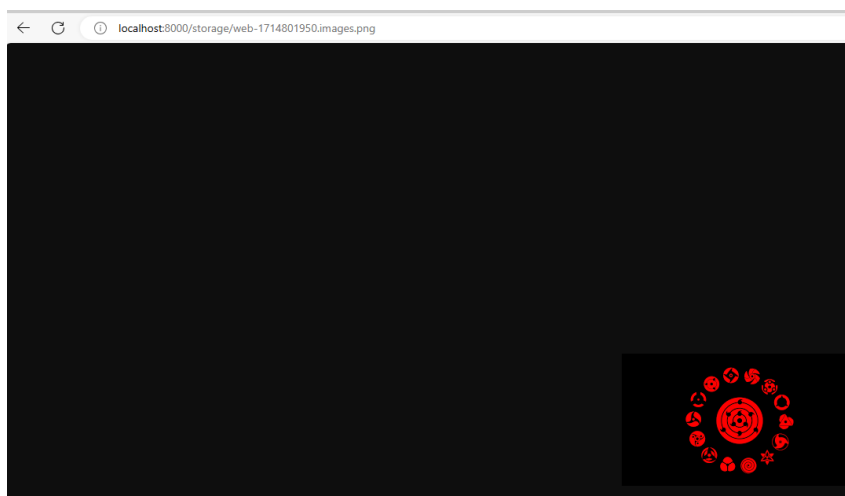
Perhatikan bahwa argumen pertama dari method storeAs() adalah 'public'(line 21). Inilah folder tempat file upload akan disimpan. Silahkan upload sebuah file, maka akan tampil hasil berikut:



← ↻ ⓘ 127.0.0.1:8000/file-upload

proses upload berhasil, data disimpan pada:public/web-1714801950.images.png

Proses upload berhasil, dan file tersebut ada di storage\app\public\ web-1714801950.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/ web-1714801950.images.png](http://localhost:8000/storage/web-1714801950.images.png)



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

```
FileUploadController.php x
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\Http\Controllers\FileUploadController > prosesFileUpload()
7 class FileUploadController extends Controller
10 public function fileUpload(){
12
13 }
14
15 7 references | 0 overrides
16 public function prosesFileUpload(Request $request){
17
18     $request->validate([
19         'berkas'=>'required|file|image|max:500',]);
20     $extfile=$request->berkas->getClientOriginalName();
21     $namaFile='web-' .time().".".$extfile;
22     $path = $request->berkas->storeAs('public',$namaFile);
23
24     $pathBaru=asset('storage/'.$namaFile);
25     echo "proses upload berhasil, data disimpan pada:$path";
26     echo "<br>";
27     echo "Tampilkan link:<a href='$pathBaru'$>$pathBaru</a>";
28
29 }
30 }
31
32
```



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`



```
FileUploadController.php X
Materi > jobsheet12 > app > Http > Controllers > FileUploadController.php > App\Http\Controllers\FileUploadController > prosesFileUpload()
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  24 references | 0 implementations
8  class FileUploadController extends Controller
9  {
10     //
11     7 references | 0 overrides
12     public function fileUpload(){
13         return view('file-upload');
14     }
15
16     7 references | 0 overrides
17     public function prosesFileUpload(Request $request){
18
19         $request->validate([
20             'berkas'=>'required|file|image|max:500',]);
21         $extfile=$request->berkas->getClientOriginalName();
22         $namaFile='web-' .time().".$extfile";
23
24         $path = $request->berkas->move('gambar',$namaFile);
25         $path =str_replace("\\","/", $path);
26         echo"Variabel path berisi:$path <br>";
27
28         $pathBaru=asset('gambar/'.$namaFile);
29         echo "proses upload berhasil, data disimpan pada:$path";
30         echo "<br>";
31         echo "Tampilkan link:<a href='$pathBaru'>$pathBaru</a>";
32     }
33 }
34
```

← ↻ ⓘ 127.0.0.1:8000/file-upload

Variabel path berisi:gambar//web-1714803121.images.png
proses upload berhasil, data disimpan pada:gambar//web-1714803121.images.png
Tampilkan link:<http://127.0.0.1:8000/storage/web-1714803121.images.png>

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



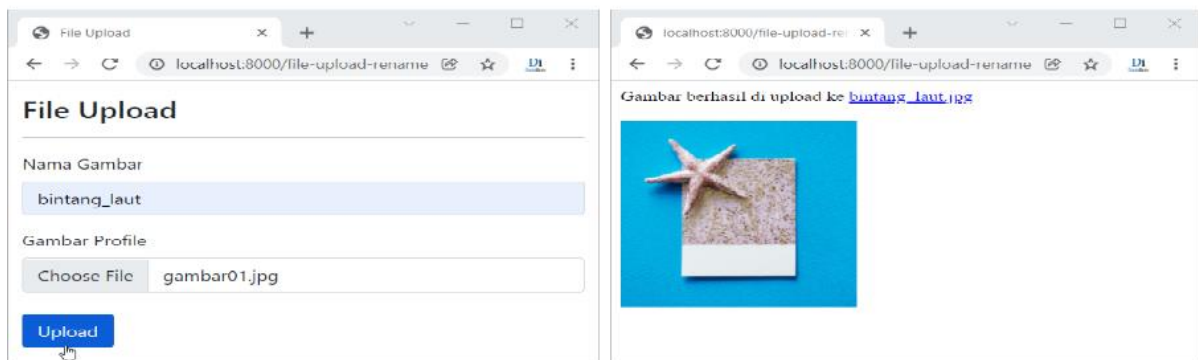
KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>

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 .



2. Tambahkan proses upload image dan application pada starter code

*** Sekian, dan selamat belajar ***