1.14

TEMPS 1 · LE PORTFOLIO · SÉANCE 4

Téléverser une image

Thibault Six · IFOSUP Wavre · 2026-2027

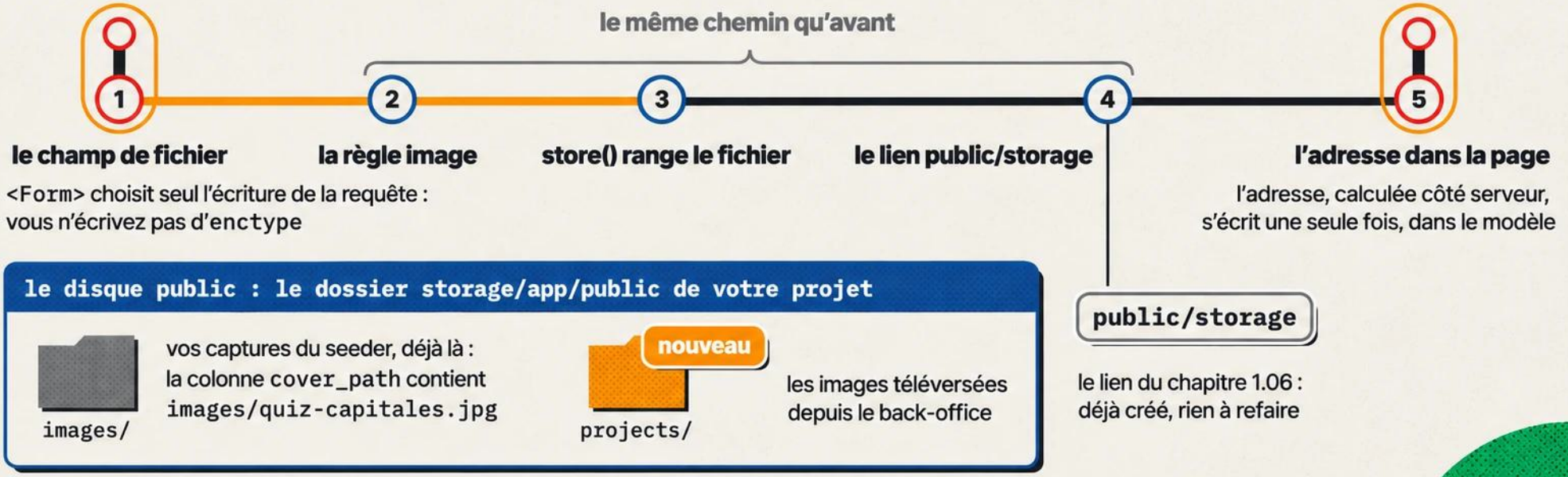
5XDYN · Projet web dynamique · 1.14



01

? Une image choisie dans un formulaire finit affichée dans une page. Par où passe-t-elle ?

Une image passe par un champ, une règle, un dossier du disque public et une adresse : deux étapes changent ici.



Suivez la ligne de gauche à droite. Les deux stations cerclées changent dans ce chapitre ; le disque public reçoit un second dossier.

1.14

? Que faut-il ajouter au formulaire de projet pour pouvoir choisir une image ?

Un Input de type file suffit : accept montre d'abord les images, mais seul le serveur vérifie le fichier.

resources/js/pages/admin/projects/Create.vue — extrait

```
<div class="grid gap-2">
    <Label for="cover">Image de couverture</Label>
    <Input id="cover" name="cover" type="file" accept="image/*" />
```

le champ dans le navigateur



`type="file"` — le composant `Input` de shadcn-vue accepte `type="file"` : ses classes prévoient déjà le bouton de choix

`accept="image/*"` — demande à la fenêtre de choix de montrer d'abord les images

`accept` ne protège rien : n'importe qui peut choisir un autre fichier, et c'est le serveur qui vérifie

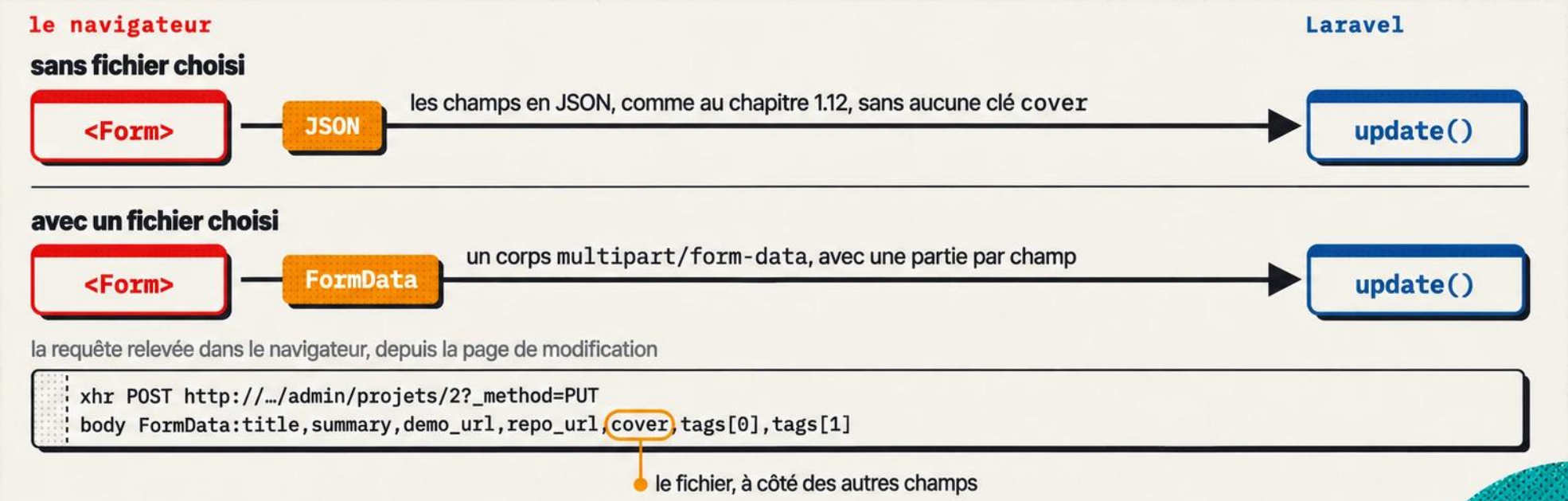
ajouté avant le `<fieldset>` des technologies ; pas d'`enctype` : `<Form>` regarde ce qu'il envoie

Trois lignes de code, un champ dans la page. L'attribut accept aide la personne qui choisit ; il n'arrête personne.

1.14

? Vous n'avez écrit aucun enctype. Comment le fichier part-il quand même vers le serveur ?

<Form> regarde ce qu'il envoie : du JSON sans fichier, un FormData dès qu'un fichier est choisi.

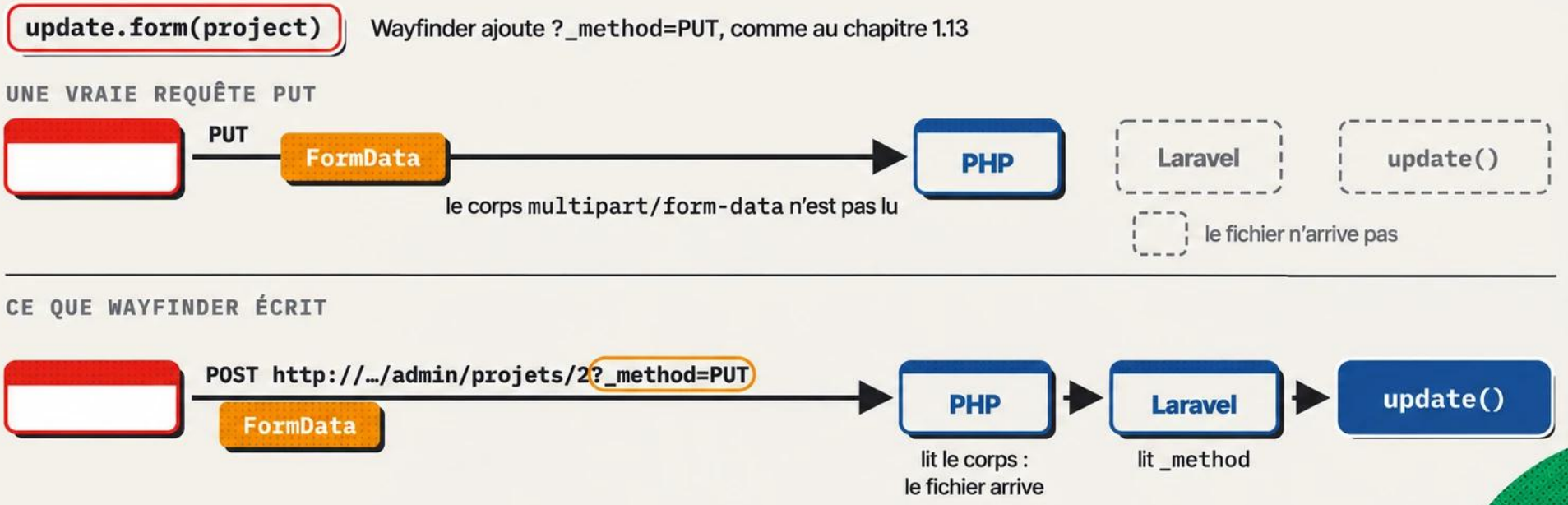


Même formulaire, deux envois. Dès qu'un fichier est choisi, <Form> change la forme du corps, sans rien vous demander.

1.14

? Modifier un projet appelle update(), une route PUT. Pourquoi la requête part-elle en POST ?

PHP ne lit pas le corps multipart/form-data d'une vraie requête PUT : elle part en POST, puis Laravel lit _method.



En haut, la vraie PUT perd le fichier en route. En bas, le POST le livre, et _method mène quand même à update().

? accept ne protège rien. Qu'est-ce qui refuse un PDF, ou une capture trop lourde ?

Quatre règles vérifient l'image côté serveur : image lit le contenu, mimes limite les formats, max compte en kilo-octets.

app/Http/Requests/ProjectRequest.php · rules()

```
'cover' => ['nullable', 'image', 'mimes:jpg,jpeg,png,webp', 'max:2048'],
```

(1) (2) (3) (4)

nullable
le champ peut rester vide

image
Laravel lit le début du fichier et y reconnaît une image

mimes:jpg,jpeg,png,webp
seuls ces quatre formats passent : `image` seule accepterait un BMP ou un HEIC

max:2048
pour un fichier, max compte en kilo-octets : 2 Mo au plus

un PDF refusé, le message sous le champ

Choisir un fichier fiche-projet.pdf

Le champ image de couverture doit être une image.

La taille du fichier de image de couverture ne peut pas dépasser 2048 kilo-octets.

« de image » : Laravel-Lang ne fait pas l'élision

« Le fichier du champ image de couverture n'a pu être téléversé. » : PHP a écarté le fichier avant Laravel, car il dépasse `upload_max_filesize`, la limite de `php.ini`, 2 Mo par défaut

Chaque règle a son libellé, chaque refus remonte à sa règle. Le dernier message ne vient pas de Laravel : PHP a refusé le fichier avant lui.

1.14

? La colonne cover_path attend un chemin. Comment le fichier envoyé devient-il ce chemin ?

store() copie le fichier dans projects/ sous un nom tiré au hasard, et renvoie le chemin qui va dans cover_path.

app/Http/Controllers/Admin/ProjectController.php · store() — extrait

```php
$data = $request->safe()->except(['tags', 'cover']);

if ($request->hasFile('cover')) {
    $data['cover_path'] = $request->file('cover')->store('projects', 'public');
}
```

except(['tags', 'cover'])
retire les deux clés que la table `projects` ne connaît pas

hasFile('cover')
vaut `true` quand un fichier est arrivé sans erreur



Lisez le code, puis suivez la ligne : le fichier entre à gauche, seul son chemin arrive dans la base.

La page a besoin d'une adresse, la base ne garde qu'un chemin. Où calculer cette adresse ?

L'accesseur coverUrl() calcule l'adresse dans le modèle : les pages lisent $project->cover_url comme une colonne.

app/Models/Project.php

```php
/**
 * @return Attribute<string|null, never>
 */
protected function coverUrl(): Attribute
{
    return Attribute::get(fn () => $this->cover_path
        ? Storage::disk('public')->url($this->cover_path)
        : null);
}
```

une méthode qui renvoie un `Attribute` : un accesseur ; `coverUrl()` donne `$project->cover_url`

`Attribute<ce qu'on lit, ce qu'on écrit>` : `string|null` à la lecture, `never` à l'écriture ; pas de mutateur, l'adresse ne s'assigne jamais

`images/meteo-wavre.jpg` cover_path

`Storage::disk('public')->url()` part de APP_URL dans .env : http://portfolio.test

`http://portfolio.test/storage/images/meteo-wavre.jpg`

cover_url, lu dans Tinker

`'cover_url' => $project->cover_url,` index() et show()

`'cover_url'` dans le only() de edit()

la page reçoit cover_url, jamais cover_path



Si les images sont cassées, regardez d'abord APP_URL.

Le calcul s'écrit une fois, à gauche. À droite, suivez un projet du seeder : son chemin devient une adresse, puis une prop de la page.

1.14

? Vous envoyez une nouvelle image pour un projet. Que devient l'ancien fichier sur le disque ?

update() supprime l'ancien fichier avant de ranger le nouveau, seulement si son chemin commence par projects/.

app/Http/Controllers/Admin/ProjectController.php · update() extrait

```php
if ($request->hasFile('cover')) {
    if ($project->cover_path && str_starts_with($project->cover_path, 'projects/')) {
        Storage::disk('public')->delete($project->cover_path);
    }

    $data['cover_path'] = $request->file('cover')->store('projects', 'public');
}
```

vaut `true` quand le chemin commence par `projects/`, le dossier des téléversements

storage/app/public/

images/

quiz-capitales.jpg ✓

une capture du seeder : jamais supprimée

projects/

~~4ayJrdZUcgdVpkLWFkKDnh9KJ9zxhzsP2pfcjaGy.jpg~~

une seconde image pour ce projet supprime ce fichier et prend sa place

sans nouveau fichier, `$data` n'a pas de clé `cover_path` : `update()` ne touche pas à l'image

`destroy()` supprime aussi le fichier, avec la même condition, juste avant `$project->delete();`

Le premier if attend un nouveau fichier, le second protège les captures du seeder. Seul un fichier de projects/ disparaît.

1.14

Les captures du seeder et les images téléversées partagent un disque. Lesquelles vont dans Git ?

images/ fait partie de votre code et va dans Git ; projects/ appartient au site en marche, comme les lignes de la base.

le disque public : storage/app/public/

	images/	projects/
CE QU'IL CONTIENT	les captures du seeder	les images téléversées depuis le back-office
DANS GIT	oui, grâce au `.gitignore` du chapitre 1.06	non
SUPPRIMÉ PAR UPDATE() ET DESTROY()	jamais	oui, quand l'image change ou que le projet disparaît

un clone de votre dépôt les retrouve après `migrate:fresh --seed`

```
$ php artisan tinker --execute="echo App\Models\Project::firstWhere('slug', 'meteo-wavre')->cover_path;"
images/meteo-wavre.jpg
```

le seeder ne change pas : `cover_path` reste un chemin relatif au disque, et le même accesseur calcule les deux adresses

Une colonne par dossier, une ligne par question. Ce qui vient de votre code se versionne ; ce qui vient du back-office vit avec la base.

? Depuis une connexion lente, l'envoi d'une image prend du temps. Comment montrer qu'il avance ?

Pendant l'envoi d'un fichier, <Form> remplit progress, et la balise <progress> dessine la barre de 0 à 100.

```
v-slot="{ errors, processing, progress }"
```

ajouté aux variables du slot de <Form>

resources/js/pages/admin/projects/Create.vue extrait

```
<progress
    v-if="progress"
    :value="progress.percentage"
    max="100"
    class="w-full"
>
```

progress null tant que rien n'est envoyé

progress.percentage de 0 à 100 pendant l'envoi

la balise <progress> dessine la barre





le bouton Enregistrer, désactivé pendant l'envoi

sur votre ordinateur, la barre apparaît à 100 : pour la voir avancer, choisissez un débit lent dans l'onglet Réseau

pas la fine barre grise du haut de la page : celle-là vient de la clé `progress` de `resources/js/app.ts`

La variable du slot passe de null à un pourcentage, et la balise HTML la dessine. Pendant ce temps, le bouton attend.

1.14

? Votre portfolio partira chez OVH au chapitre 1.22. Qu'est-ce qui pourrait y casser vos images ?

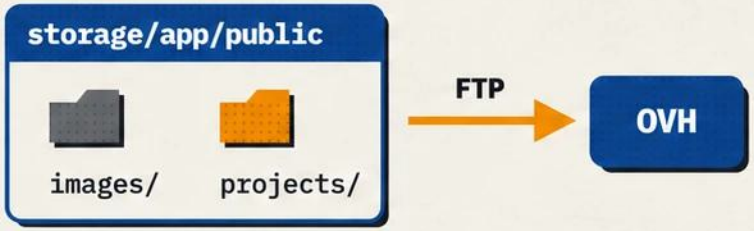
Chez OVH, trois points : pas de lien symbolique, un APP_URL public, et storage/app/public envoyé avec la base.

1 ~~public/storage~~

l'hébergement mutualisé d'OVH ne suit pas les liens symboliques : le chapitre 1.22 change la racine du disque sur le serveur

2 **APP_URL**

l'adresse publique de votre site, sinon chaque `cover_url` pointe vers `portfolio.test`

3 

le seeder ne tourne pas chez OVH : sans ce dossier, les chemins de la base désignent des fichiers absents

Trois points à garder pour la mise en ligne, détaillés au chapitre 1.22. Rien à faire aujourd'hui.

1.14

? Qu'est-ce qui doit être vrai à la fin ?

À vous, trente minutes : une couverture par projet, rangée, remplacée et supprimée sans fichier orphelin.

À FAIRE

1. La règle `cover` et son nom dans `ProjectRequest`.
2. `store()`, `update()` et `destroy()` qui rangent, remplacent et suppriment le fichier, ce dernier seulement dans `projects/`.
3. L'accesseur `coverUrl()` dans `Project`, utilisé par `index()`, `show()` et `edit()` à la place des appels à `Storage::disk('public')->url()`.
4. Le bloc « Image de couverture » dans `Create.vue` et `Edit.vue`, avec `progress` dans le slot et l'aperçu dans `Edit.vue`.

À VÉRIFIER

30 min

- [ ] `portfolio.test/projets` montre vos trois captures, et le code source de la page donne des `cover_url` en `http://portfolio.test/storage/images/…`.
- [ ] Un nouveau projet créé avec une image l'affiche sur sa fiche, avec une `cover_url` en `http://portfolio.test/storage/projects/…`.
- [ ] Remplacer l'image d'un projet du seeder garde la capture dans `storage/app/public/images/`.
- [ ] Remplacer une seconde fois l'image d'un projet laisse un seul fichier pour ce projet dans `storage/app/public/projects/`.
- [ ] Modifier un projet sans choisir de fichier garde son image.
- [ ] Un PDF est refusé avec un message sous le champ.

Quatre étapes à gauche, six vérifications à droite ; les deux remplacements se vérifient dans le dossier storage/app/public/.

