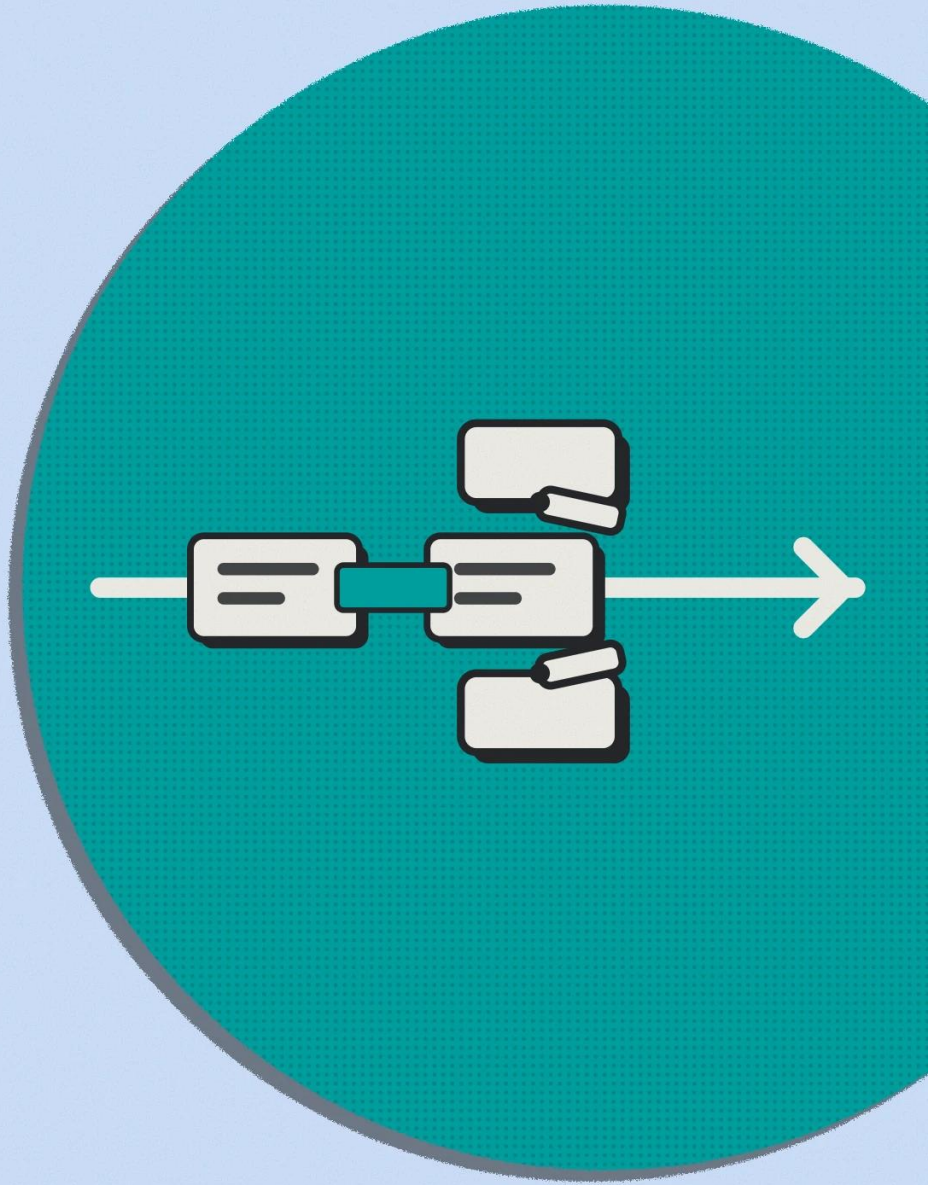
2.03

TEMPS 2 · LE TAMAGOTCHI TROUÉ · SÉANCE 1

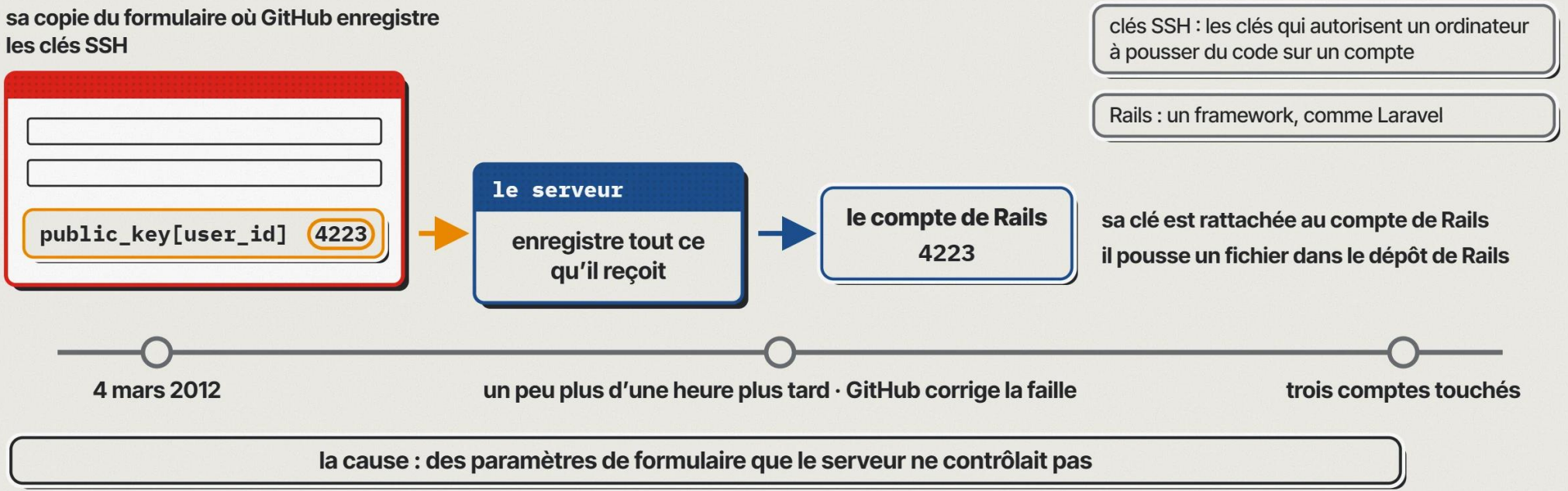
L'assignation de masse



Thibault Six · IFOSUP Wavre · 2026-2027

? En 2012, comment un développeur a-t-il pu écrire dans le dépôt de Rails sans en avoir le droit ?

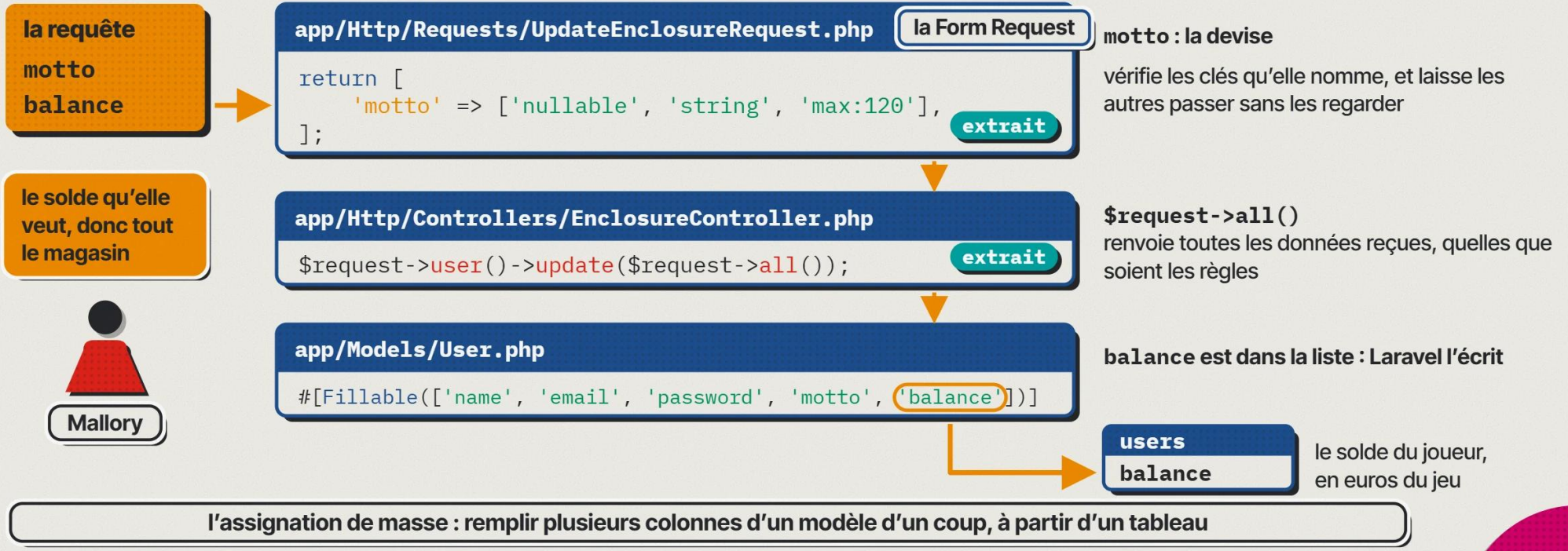
Homakov glisse un champ caché dans le formulaire, et GitHub enregistre tout ce qu'il reçoit.



Homakov n'a modifié que sa copie du formulaire, dans son propre navigateur ; c'est le serveur de GitHub qui a accepté le champ en trop.

? La devise est validée. Si la requête porte aussi balance, qu'est-ce qui arrive dans update() ?

La validation ne regarde que `motto` ; `all()` transmet aussi `balance`, et `#[Fillable]` l'accepte.

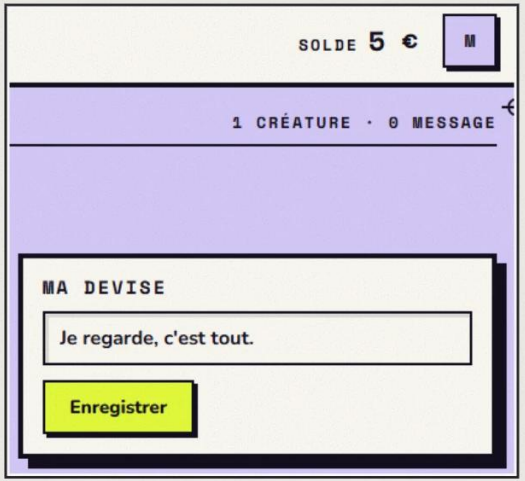


Suivez `balance` de haut en bas : la règle ne le regarde pas, `all()` le rend, `#[Fillable]` l'accepte. Rien ne l'arrête.

? Le formulaire n'a qu'un champ. Comment Mallory y ajoute-t-elle balance, sans aucun outil en plus ?

Dans l'onglet Éléments, Mallory ajoute un champ `balance` au formulaire, juste après le bouton.

AVANT



`/enclos/2`, l'enclos de Mallory

F12, ONGLET ÉLÉMENTS

le formulaire de la devise, coupé à l'essentiel

```
<form action="/enclos?_method=PATCH" method="post">
    <label for="motto">Ma devise</label>
    <input id="motto" name="motto">
    <button type="submit"> Enregistrer </button>
</form>
```

```
<input name="balance" value="1000">
```

clic droit sur `<button type="submit">`, « Edit as HTML »

APRÈS



Mallory

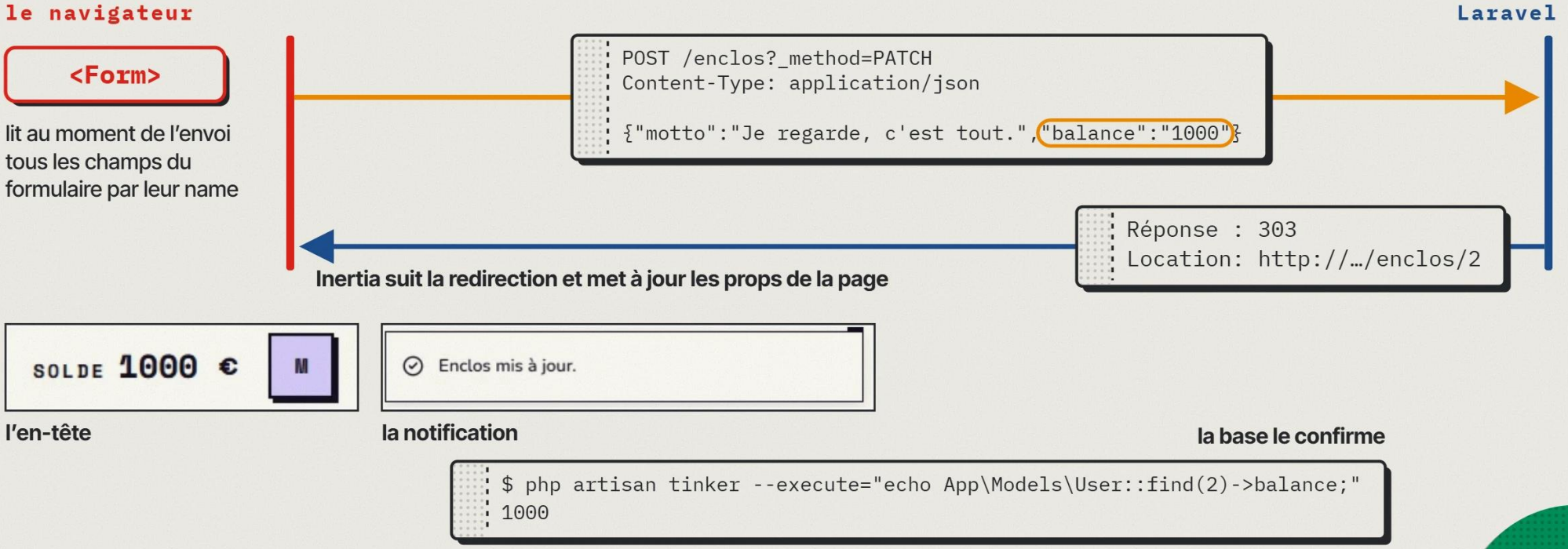
le nouveau champ, sous le bouton, sans bordure



À gauche, la page telle que le serveur l'a envoyée ; à droite, la même page modifiée dans le navigateur de Mallory.

? Mallory clique sur « Enregistrer ». Le champ ajouté part-il avec la devise ?

Le composant <Form> envoie tous les champs présents au clic : le solde passe à 1000 €.



La page affichait un champ, la requête en porte deux. Le serveur accepte tout, et l'en-tête affiche le nouveau solde.

? Sur votre copie, l'attaque réussit-elle, et que dit le test de la faille avant toute correction ?

À vous, quatre minutes : refaites l'attaque sur votre copie, puis lisez le test rouge.

À FAIRE — 4 min

1. `php artisan migrate:fresh --seed`
2. Connectez-vous comme Mallory et ouvrez « 02 Enclos ».
3. Dans l'onglet Éléments, ajoutez `<input name="balance" value="1000">` juste après le `</button>`.
4. Ouvrez l'onglet Réseau, puis cliquez sur « Enregistrer ».
5. `php artisan test --filter=Flaw03`

le test crée son propre joueur, avec 50 € : ce n'est pas la Mallory du seeder

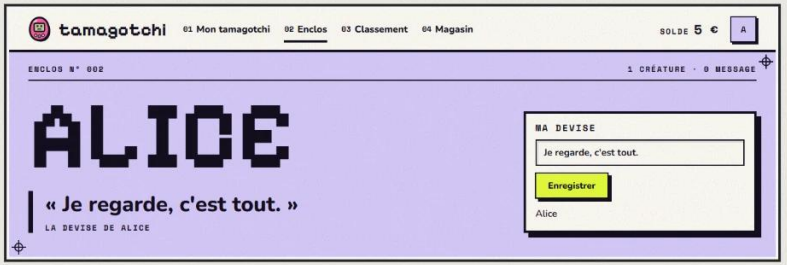
un champ hors de la balise n'est pas lu

À VOIR

- [ ] l'onglet « Charge utile » de la requête `enclos?_method=PATCH` : deux clés
- [ ] le solde de l'en-tête

```
  × extra fields sent with the motto are ignored                0.66s
-     'name' => 'Mallory',
+     'name' => 'Alice',
…
-     'balance' => 50.0,
+     'balance' => 1000.0,
```

- avant l'envoi
+ après l'envoi



essayez aussi un champ name qui vaut Alice, et regardez le titre de l'enclos

Cinq étapes à gauche ; à droite, ce que vous devez voir avant de corriger, dont le test rouge.

? La Form Request connaît la règle de motto. Comment ne garder que les clés que ses règles nomment ?

Avec `validated()`, seules les clés qui ont une règle restent : `balance`, sans règle, disparaît.

VOUS DEVEZ OBTENIR

Le formulaire de la devise ne change que la devise, quoi que la requête contienne d'autre.

10 min — cherchez dix minutes, puis ouvrez la correction

app/Http/Controllers/EnclosureController.php · update()

```
$request->user()->update($request->all());
```

la ligne à remplacer

au chapitre 1.12, `store()` ne passait jamais `all()`
`$request->safe()->except('tags')`

CE QUE GARDE validated()

le portillon : la règle de validation



Le tableau qui entre porte deux clés ; celui qui sort n'en garde qu'une, celle que la règle nomme.

? Si balance sort de #[Fillable], comment le magasin et les codes cadeau changent-ils le solde ?

Hors de #[Fillable], balance est ignoré sans erreur : le solde s'écrit par affectation.

Le solde ne peut plus venir d'aucun formulaire, même par erreur dans un autre contrôleur.

app/Models/User.php

```php
#[Fillable(['name', 'email', 'password', 'motto', 'balance'])]
```

deux contrôleurs changent le solde avec `update(['balance' => …])`

BALANCE RETIRÉ DE LA LISTE

```
App\Models\User::find(1)->update(['balance' => 1000])
true
App\Models\User::find(1)->balance
120
```

une affectation : `$user->balance = …`, suivie de `save()`

`update()` répond `true`, et le solde n'a pas bougé

Un achat ne coûterait plus rien, et un code cadeau ne créditerait plus rien.

validated() protège ce formulaire-ci. #[Fillable] protège toutes les méthodes qui passeront un jour un tableau reçu à update() ou à create().

Retirer `balance` de la liste ferme la porte aux formulaires, mais aussi au magasin : le serveur écrit donc le solde lui-même.

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

À vous : le test passe au vert, et la même attaque laisse le solde de Mallory à 5 €.

À FAIRE 5 min

1. `php artisan test --filter=Flaw03`
2. `php artisan test` : de 19 tests rouges à 18
3. `php artisan migrate:fresh --seed`, puis reconnectez-vous comme Mallory
4. de nouveau `<input name="balance" value="1000">`, puis « Enregistrer »

À VÉRIFIER

```
 PASS  Tests\Feature\Flaws\Flaw03MassAssignmentTest
✓ extra fields sent with the motto are ignored

Tests:    1 passed (3 assertions)
```

la requête porte toujours `"balance":"1000"`, mais le solde reste à 5 €





- [ ] un « Bonbon » pour Encre : SOLDE 4.9 € ; le code cadeau 4821 : SOLDE 9.9 €

DANS VOTRE CARNET DES FAILLES

Un contrôleur ne passe à `create()` ou à `update()` que des données validées, jamais `$request->all()`, et `#[Fillable]` ne liste que des champs qu'un formulaire a le droit de remplir.

Quatre étapes à gauche ; à droite, le test vert, l'attaque qui échoue, l'achat et le code cadeau qui marchent encore.